.EXE

MAY 1991 VOL 5 ISSUE 11

NOW WITH BLUE ONES !

The Software Developers' Magazine

Our theme this issue: Graphics.

How they bring chocolate to life.
Al Roth on the making of animated commercials.

Are your graphics too flat?
We begin a practical series on 3D.

A treat for Turbo Pascal users:
We look at a new Windows version.

All about the bug that slept for 150 years -
We salute Babbage, the British pioneer.

Vol 5 Issue 11

.EXE

May 1991

Thanks to Snapper Bytes and Rowntree Mackintosh Confectionery Ltd for permission to use the cover photograph.

General

.EXE Magazine is independent and not affiliated to any vendor of hardware, software or services. It is published by Process Communications Ltd, 10 Barley Mow Passage, Chiswick, London W4 4PH.
Tel: (Advertising/Editorial/Production) 081 994 6477
(Subscriptions) 044282 4501
Facsimile: 081 994 1533 Telex: 8811418 SPACES G
ISSN: 0268-6872

Subscriptions

.EXE Magazine is a monthly journal for software developers. It is available only by subscription, at a cost of £35 per annum (11 issues) in the UK (see insert card for academic and overseas rate). 'A subscription implies that this journal will be sent to the subscriber until one of the three expires' - AG Macdonell. The magazine is published around the 26th of the month preceding the cover date. There is no January issue. To subscribe, please call 044282 4501, or write to The Subscriptions Manager, .EXE Magazine, 10 Barley Mow Passage, Chiswick, London W4 4PH. We can invoice your company or take ACCESS and VISA cards. In case of a query about your subscription, please call 044282 4501.
Back issues are available at a cost of £3.50 each - call our Chiswick office (081 994 6477) for a list of issues still in print. We also offer a range of magazine binders, disk holders etc - please call for details.

Editorial

Editorial enquiries should be addressed to The Editor, .EXE Magazine, 10 Barley Mow Passage, Chiswick, London W4 4PH. We welcome letters, opinions, suggestions and articles from our readers. If you are interested in contributing articles, please write to this office for a copy of our Contributors' Guide.
Information contained in .EXE is believed to be correct. If errors are found, we will endeavour to publish a clarification in the next available issue.
From time to time, we offer to copy the PC program code described in an article onto diskette for our readers. In this case, please send a blank, formatted disk with a self-addressed, prepaid mailer to the editorial address given above. We can copy both 5.25" and 3.5" disks.
The publishers can accept no liability for any consequences of using software distributed in this way.

Advertising

If you are interested in advertising in this magazine, please write to the Display Advertising Manager/Recruitment Sales Manager (as appropriate), .EXE Magazine, 10 Barley Mow Passage, Chiswick, London W4 4PH, or call 081 994 6477 for details of our advertising rates.

Pronunciation

The name of .EXE Magazine is pronounced to rhyme with 'not sexy magazine'.

Copyright

Issue theme: Graphics

A Paradigm too far

Jim Cooling pleas for good, simple English in computing articles.

Last August, .EXE published an article by John Daniels concerning OOP and C++. One reader, R.M. Coleman, was sufficiently agitated by this to write to the Editor. The issue? No, not OOP, not even C++, but the use of the word 'paridigm' *(sic)*. Now, Mr Coleman's criticism of the use of obscure and misspelled words (it should have been *paradigm*) seemed very reasonable to me. That might have been the end of it - except for the put-down tone of the Editor's reply. My long-term irritation at the abuse of the English language by computer specialists finally boiled over. Hence this article.

The seeds of my annoyance were planted many years ago when I taught myself Coral 66 from a compiler manual. For sheer dullness, verbosity and incomprehensibility this document was matched only by the literary section of the Sunday Times. It induced symptoms akin to Guinness fatigue after a long weekend in Dublin. I soon discovered that such language was the norm, not the exception, in computing science. New words entered my vocabulary: denotational semantics, orthogonal design, instantiation, paramodulation. Many were undefined in either the Oxford Reference Dictionary or the Oxford Dictionary of Computer Science. But that didn't seem to inhibit their use. And then, a few years ago, came the crackerjack of them all, the paradigm. This, which **is** defined in the OED, is one of the most abused words in computer science. The dictionary says it is a noun. I have seen it used - frequently out of context - as an adjective and an adverb. If it doesn't go out of fashion soon, it's sure to be verbed.

Before we continue with the paradigm saga, a brief digression. How have we got into this sorry state of affairs? There are (at least) three identifiable reasons for the increased use of mind-numbing computerese:

The cultural influence of 'good' literature.

The wish by writers to demonstrate their superior intellect to the reader.

The ceaseless quest for academic respectability.

Consider the first point. What is our definition of a 'good' text? Well, in our business, it is one which should convey information clearly, simply and accurately. It is a bonus if it is also entertaining. Can we say the same about, for instance, *Sons and Lovers*? It may be entertaining (subjective); but it certainly doesn't meet our other criteria. But then it was never meant to - the rules of classical literature are inappropriate for technical literature.

But there are plenty of pedantic bores about, just waiting to call us to heel if we dare to use good, clear **ordinary** words. Consider the following quotations from a recent national newspaper article entitled 'Debasing print to cater for idiots'. 'This demeaning of the text is endemic in publishing ... to lower the level of discourse to the public's presumed ability to absorb it ... no one talked about what magazines have traditionally done, which is to educate their readers.' Shades of Lord Reith. Would you believe that the author was referring to the magazine business? Attitudes like this pervade the senior strata of science and engineering. No wonder so many technical articles are written in a turgid style.

The second point - showing off - is nothing new. It used to be almost endemic in British text books, though things have improved in recent years. The attitude of such authors was summed up beautifully by Nicholas Bagnell: 'Now it is as though critics were somehow reluctant to praise lucidity, which is perhaps only human. To say that a novel is to be commended because the common reader can understand it is a dangerous admission for those who earn their livings by explaining the otherwise inexplicable. There is no mileage in it. A fancy adjective helps restore dignity and maintain mystique, as a wig may disguise an all-too-human judge.'

And what of academic respectability? You've heard the old phrase 'publish or perish'. This is even more true today, with the advent of regular reviews and assessment of the research performance in the higher education system. To have a paper accepted for publication by a learned journal, certain criteria must be met. Clarity of presentation and comprehensibility appear not to be among these criteria. If they were, you might realise that the subject under discussion is not so difficult after all. Cynicism? Perhaps, but just have a look through some issues of software transactions and proceedings.

Back to the paradigm. My working dictionary (Oxford Reference) defines it as 'an example or pattern, especially of the inflexions of a noun, verb etc.' But its use has gone well beyond this

GREY MATTER

2 Prigg Meadow, Ashburton, Devon TQ13 7DF

TEL. (0364) 53499

GREY MATTER

2 Prigg Meadow, Ashburton, Devon TQ13 7DF

TEL. (0364) 53499

fairly simple definition. How about (from a book review) 'now that he is dead, the career begins to look like a paradigm.' You *what?* A couple of book titles: 'Paradigms Lost' and, closer to home, 'The Book Paradigm for Improved Maintenance', which is defined as 'a typographic arrangement of source code that serves as an efficient form of information organization and presentation.'

A special mention for a published article entitled 'A Compositional Approach to Multiparadigm Programming'. The journal concerned printed a letter commenting on the piece, which ended: 'Let's stop applying techno-babble to common-sense engineering practice with the apparent goal of making it inaccessible to the practitioner.' Quite so. The author's reply was either beautifully tongue-in-cheek, or it showed a complete loss of contact with reality. Quote: 'The article also explores the conditions under which single-paradigm validation operations (algorithmic analysis, verification, and testing of single-paradigm programs in isolation) retain their meaning in a multi-paradigm setting.'

What are the consequences of this verbal posturing? First, it sets up a barrier between the academic (mainly theoretical) community and the rest of the software fraternity. It impedes progress because practitioners either (a) don't read research-oriented journals (they find them a great turn-off), or (b) do read such journals but can't understand the contents.

Second, it creates a barrier between developers of software and users. The consequences of this should not be underestimated. In a recent article, Anthony Sampson wrote: 'It's in computers that the engineers and programmers have won their greatest victories in defeating the public... Their trick has been not just to use incomprehensible language, but to use ordinary words in a quite different sense... It's the work of a tribe with a fierce grievance against the public... the technologists are trapped in their own specialization, which they relish and guard against the larger world.' Enough said.

The situation can be improved dramatically if a few straightforward rules are followed. There's nothing new here; just see Lee Harrisberger's 1966 book 'Engineermanship - a philosophy of design'. Use simple words instead of complex or obscure ones. Choose words which are in common use. If you must use a less well-known word, make sure you understand its meaning. Write in a clear, direct style. Get a second opinion on your work. And always remember that the reader, not the writer, is the one who really counts.

I'd like to leave you with something that, most definitely, should not be read on the morning after the night before. It comes from a well-known language reference manual, and is intended to define a specific aspect of the language:

'If a subprogram declaration, a package declaration, a task declaration, or a generic declaration is a declarative item of a given package specification, then the body (if there is one) of the program unit declared by the declarative item must itself be a declarative item of the declarative part of the body of the given package.'

Good writing.

EXE

Jim Cooling is a senior lecturer in the Department of Electronic and Electrical Engineering, Loughborough University of Technology. The opinions of students concerning his writing and lecturing style are not available at this time, but may be known once the final year papers have been marked.

New version of Brief
*A new version of Brief, the **very** popular programmers' editor, is now shipping. V3.1 introduces Microsoft mouse support, tiled windows, EMS support and a redo facility. Brief is available from Solution Systems for £199 + VAT on 0763 244141.*

Protected mode DataWindows
DataWindows, the windowing library from Greenleaf, is now available for protected mode apps. The library requires Watcom C 386 V8.0 or MetaWare High C V2.3 and the Phar Lap 386 extender. It costs $386. Greenleaf is on 0101 214 248 2561.

Practical LISP
The proceedings of the recent European Conference on the Practical Applications of LISP, held in Cambridge this March, are now available. Systems described in detail range from a school timetabling system to a scheduler for an entire airline. It includes papers by Dick Gabriel, Luc Steels and Gregor Kizcales. The publication costs £24.95, and can be ordered from EUROPAL on 0306 77331.

SQL OS/2 Code
GPF is a GUI code generator for Presentation Manager that allows you to build in SQL code. So, for example, you can paint a dialog box that will automatically fill up with a SQL extracted table at run-time. Multi-threaded SQL requests are supported. Database handling is performed via the Extended Edition DBMS, so you'll need an OS/2 EE, the PM SDK and 6 MB of memory. Microformatic GPF costs £1895 + VAT. More information from QA Training on 0285 655888.

European C++ User Group
Mike Banahan has launched a user group catering for the needs of European C++ users. The membership cost is £50 for individuals, which includes a newsletter and cut-price conference tickets. The group will also hold biannual forums in European cities; the first will be in London this September. Interested? ECUG is on 071 253 5121.

Smalltalk/V for Windows
Cocking and Drury is now taking orders for Smalltalk/V for Windows. The class library is almost identical with that of its PM product, but with class additions for matters such as DDE support - and different system API support, of course. Smalltalk/V costs £320; Cocking and Drury is on 071 436 9481.

OS/2 Help

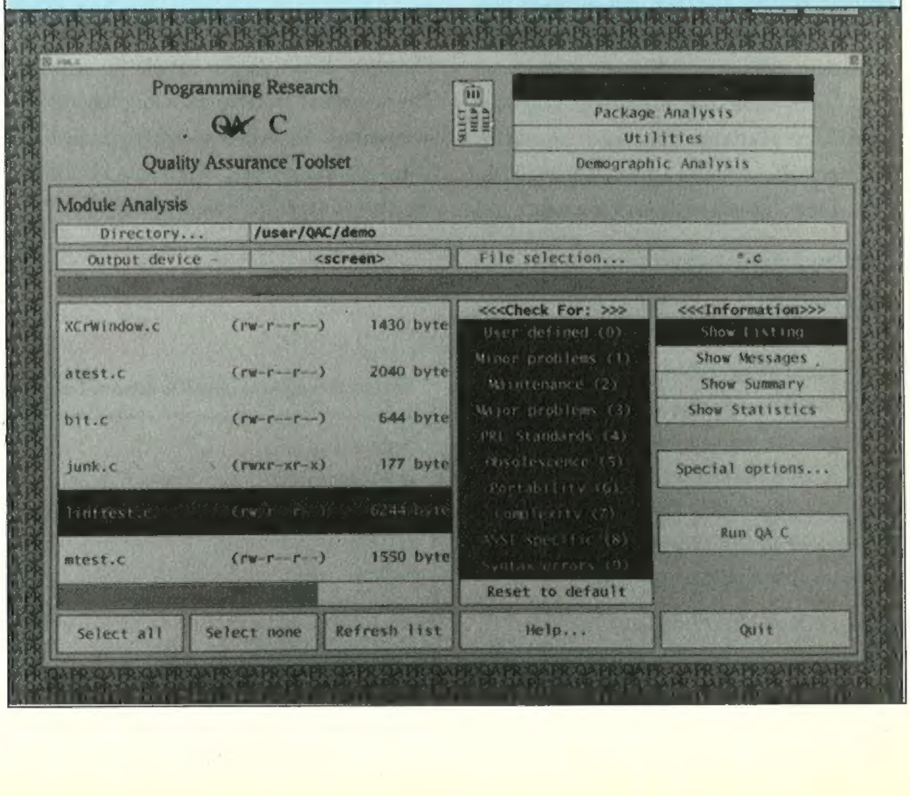
Microsoft has launched two products for OS/2 users. The first is an SDK that helps build server applications that are compatible with, and can interface to, the Microsoft OS/2 SQL Server. The package is called the SQL Server Gateway Toolkit, and contains a library of routines that handle DB-Library calls from a SQL-Server client (DB-Library is the SQL Server API). Microsoft intends the package to be used to link data held in the SQL Server RDBMS with remote databases. The package can support up to 50 simultaneous client connections.

Also from Microsoft is a new rationalisation of the horrific morass that is OS/2 device driver writing. It's called LADDR (layered device driver architecture). Device drivers are now subdivided into a number of layers with the device-specific interface at the top, and hardware interfacing at the bottom. The advantage is that now Microsoft can now provide standard interfaces for popular expansion boards, which developers can customise with their own functions. Already available are interfaces for Adaptec, Future Domain, NCR and Western Digital controller cards, with more promised. Microsoft can be contacted on 0734 391123.

The mousetrap trap

The .EXE offices attract better mousetraps like Private Eye attracts writs; every week a small company offers us an *amazing* DOS un-delete utility, or a TSR which at the touch of a *single button* throws up a table of the ASCII set *twice* as fast as any competitor. Usually, we give these programs a miss, partly because we are an Evil Magazine In The Pay Of The Corporate Conspiracy, and partly because the improvement generally isn't worth the added cost.

But you can be too cynical. Programming Research's QA C is a code checker for UNIX, so it's possible to discard it as just a better lint. But testing and quality control is a priority these days and while lint is good, it's a limited freebie; there's room (and money) for a better code checker. QA C is a happy competitor for this. As well as doing all lint-ish things, it adds checks based around Tom Plum's 'C Programming Guidelines' (which it carries in on-line form) and the full ANSI C standard. It contains around 1200 warning messages but, unlike lint, doesn't present all of them whenever you miss a colon. A number of metric tests for code quality are also tied in. Quality results can be compared with a number of C application packages including averages for graphical, mathematical, and business apps (one of the comparisons included is the X11 source - and if you're below the values for that, you have some deeply sick code). It's also prettier than lint (as it should be, prices start at £15k for a 5-user system). It was compiled on a validated ANSI compiler, and Programming Research has geared the whole package around ISO 9000/BSI 5750 to help companies who are seeking quality validation on those standards. PR also does a similar product for FORTRAN users - for more information on both applications, call 0372 62130.



Sycero adds Clipper support
System C, maker of the database applications generator Sycero dB, has added Clipper 5.0 support to the product. The new version - V2.55 - is available free of charge to V2.5 users. Other Sycero dB users can get it for £195; RRP is £595.

FORTRAN 90 for 386
Lahey has released version 4.0 of its F77L-EM/32 DOS-extended FORTRAN compilers. The latest version includes more FORTRAN 90 additions: ALLOCATABLE, ALLOCATE and DEALLOCATE, SELECT CASE, CASE, and END SELECT, CYCLE and EXIT. Users porting from the VAX will have '$' and 'Q' formats for VAX FORMATting. Lahey products are sold in this country by System Science (071 833 1022).

CASE:W goes corporate
CASE:W, the Windows code generator, now has a Corporate big brother. CASE:W V3.10 Corporate costs £995 (a £200 upgrade away from CASE:W classic). It can now check for CUA compliance, and supports secondary windows, including a full MDI interface with icons and so on. Tacking on your own message handlers is easier, too. It's available from QA Training on 0285 655888.

dBASE for Suns
dBASE IV is now available for Sun UNIX. It requires 4 MB of RAM, and costs from £995. Distributors in this country are Frontline (0256 463344) and Technology plc (0925 30404).

Leave it to ABTA
Micronet, the microcomputer information service on Prestel, has changed its tariffs to allow Micronet users free access outside of business hours. It was the introduction of off-peak usage charges a few years back that lost the service many of its original readers. Now, with the rise of services like CIX, and the albatross of Prestel's outmoded display format (it sends data in a 40x25 text format, for example, because that's all late '70s tellys could manage), Micronet's repentance seems very little and rather late.

Modula conference
The Second International Modula-2 Conference will be at the Loughborough University of Technology, September 11-13. The committee is still examining the submitted papers, so no agenda is available yet, but if last year's event in the unfortunately named Bled, Yugoslavia is anything to go by, it should be both interesting and influential. Those wishing to register should call 0509 222174.

TechnoJock

US-based TechnoJock Software scored an instant hit in this office with its corporate motto ('In the Software Business since Tuesday'). Its product is a large Turbo Pascal (DOS version 5.5 or greater) class library called the Object Toolkit, which is supplied as source code for $85 including postage ($20 more from June). The contents consist of interface objects (scrollable windows, menus, forms etc) and other useful bits (list manipulation, a computer hardware configuration object and so on). TechnoJock offers a more complete and flexible approach to the user interface than the Turbo Vision bundled with TP6 - for example there are Lotus-style menus as well as CUA - but it suffers from the fact that it isn't event-driven, and so is incompatible. If you aren't into Turbo Vision, then TechnoJock's number is 0101 713 493 6354. If you are, then the news is that TechnoJock is working on a TV add-on. We will tell you when it appears.

386 | DOS-Extender V3.0

The new version of Phar Lap's 386 | DOS-Extender is out. In preparation for the imminent launch of DOS 5, Phar Lap has finally implemented the XMS specification, so Phar Lap extended products can now be used in Windows standard and real. Still no sign, however, of a DPMI extender - although Extender V3.0 applications have been moved down to Ring 3 in preparation. The new extender has a smaller footprint: it takes up 62 KB of conventional memory when linked - that compares with V2.2 which took out 160 KB. The Virtual Memory Manager bundled with the extender has been upped too, to include the mapping of data files into extended memory. A good idea, and one that Systemstar (0992 500919) will love to tell you about, because it's Phar Lap's British distributors.

Keep them CDs rollin'

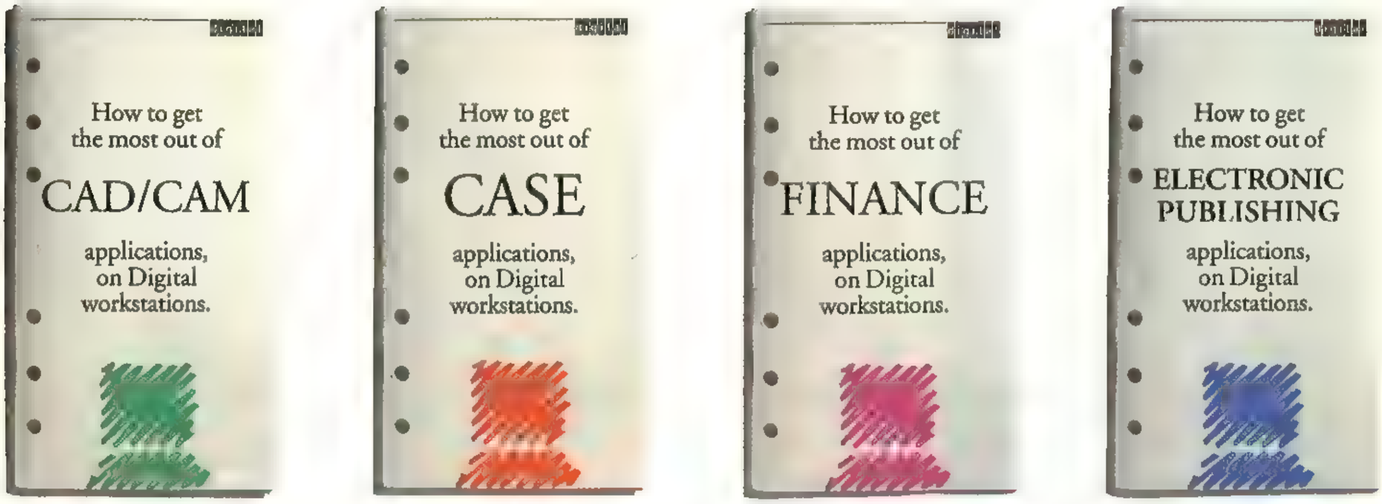
In an addendum to this month's Code Page, you might like to know that two more CD-ROM specifications have been recently submitted, and are currently being examined, by the American standards authority NIST. Both are extensions to the current High Sierra/ISO 9660-1988 format. The first, called the System Use Sharing Protocol (SUSP), sets out standards for multiple file system extensions on High Sierra disks, allowing more than one O/S version to be stored on the same CD - so a package running on many platforms can be distributed on one CD pressing. The second, whose name fits much better the Wild West flavour of previous CD standards, is called Rock Ridge, and allows UNIX files to be saved in a UNIX mountable form. Previously, UNIX applications have had to be installed from CD onto other media before running. Parties interested in reviewing the new specifications should call Robert J Niland on 0101 303 229 4014 (rjn@fc.hp.com).

New MASM announced

Microsoft has announced the release of its all-new macro assembler, MASM 6.0. Significant features of the assembler include a very fast single pass assembler and linker, and true 32-bit flat model support for the forthcoming OS/2 version 2.0. The assembler is integrated into the Programmers WorkBench (PWB), the slow but powerful heart of the Professional Development System that embraces all recent Microsoft languages. We'll have a full review soon, but briefly, MASM V6.0 includes an extensive set of simplified directives and built-in macros (see the code example for examples) that promises to transform the look of assembly language, especially when it comes to handling the ever increasing complexity of GUIs; Windows specific directives and the ability to generate a whole new range of prolog/epilogs that make the writing of Windows applications in this new assembler less of the remote possibility that has been up until now. Pricing is expected to be similar to MASM V5.1 (£115).

```
    .CODE
    .STARTUP                    ; Initialise

    .WHILE      1               ; Loop forever (or til break)
    mov         ah, 07h         ; Get key without echo
    int         21h

    .BREAK      .IF al == 13    ; Terminate if ENTER
    .CONTINUE .IF (al < '0') || (al > '9')    ; Skip if not digit

    mov         dl, al          ; Copy
    mov         ah, 02h         ; Output character
    int         21h
    .ENDW

.EXIT           0               ; Exit with return code 0
```

digital



Take guidance!

You've found your application ... you've seen it running ... and you know what it can do.

Just two more decisions to make ... what platform to use, and where to get support for the smooth running of the whole solution.

Digital makes these decisions easy. We've taken four of the most popular application areas – CAD/CAM, CASE, finance, and electronic publishing – and produced a free guide for each that covers the commonly used applications packages, and the machine and network configurations that will help you get the most out of them.

Specific topics include ...

- how to get your products to market *faster* with CAD/CAM, by integrating design engineers with the whole design team ...
- Digital's COHESION environment for CASE – for software that's successful, on time, serviceable to spec., and at the right price ...
- how finance systems can integrate applications to deliver intelligence right to the desk-top ...
- how Digital provides electronic publishing solutions that go beyond the desktop, covering the complete document management and distribution process.

Each guide covers the comprehensive support Digital offers, and is accompanied by information supplied direct by third parties.

See how they run on workstations from Digital

To match your choice of applications, Digital offers a choice of workstations – with a wide range of price/performance options. Whatever your application there's a Digital workstation to run it, from entry level VMS or UNIX® through to high-performance UNIX workstations, VMS and UNIX servers – all with unmatched capability for system expansion and integration into your enterprise.

And support? From Digital, it's comprehensive – through service, training and consultancy. In fact, support from Digital starts *before* you've even chosen a Digital workstation, with these four new Workstation Application Guides.

Send for your *free* copies – today!

Free, pocket-sized, packed with information – these are guides you should have *now*. Choose one, or ask for the whole set. For your copies post the coupon today.

To: Application Guides,
Digital Equipment Co. Ltd, PO Box 525,
RHA-PR, Maidenhead, Berks., SL6 1YU.

Please send me the application guide for:

☑ *(Tick)* ☐ CASE
☐ CAD/CAM ☐ Electronic Publishing
☐ Finance

☐ Also Digital's range of workstations

Name *(Mr, Mrs, Ms)* ____________

Position ____________

Organisation ____________

Address ____________

Postcode ________ Tel ________

Industry ____________

EXE/051

CIRCLE NO. 744

Microsoft joins OMG
Microsoft has joined the committees of the burgeoning Object Management Group. It would be churlish, of course, to suggest a connection between this and Borland/Whitewater's recent submittal of ObjectWindows as a standard class library for Windows...

Worms in the Apple
Our reviewer of Borland C++ for Windows in March admitted difficulty in finding anything nasty to say about it. Now he's discovered one: Windows applications linked under BC++ won't run in real mode. Apparently, Borland USA felt that real mode was passé. So passé, in fact, that they neglect to mention this fact anywhere in packaging or publicity, other than deep in the README.

New support for Ergo DOS-Extender
The latest version of the Ergo DOS-Extender - the TurboDrive extender packaged with Borland products - contains new support for Borland C++, Turbo C++, Metaware V1.7 and 2.31, and Microway NDP Pascal, FORTRAN and C. Library support now includes C-Analyst, C-Scape, the GSS graphics library, Halo Pro and MetaWindows. Ergo is on 0101 508 535 7510.

Grey days
The Software Construction Company, chuffed at its appointment as first official UK Borland language distributor, is offering upgrades to users who have bought 'Grey' imports at standard Borland prices. So, for example, Turbo C++ users can upgrade to Borland C++ for £99, and anyone with a competitor product to Paradox can buy Paradox for £149. The offer is valid until 31st May 1991. And if you spend over £100, you get Flight Simulator, too. TSSC is on 0763 244114.

Standard on standards
The BSI (0908 220022) has published a standard on recommendations for the achievement quality in software, VS 7165. The standard covers specifications and plans, codes of practice, and quality control, and advises on how to achieve set standards. It costs £35.40 .

Contacting us
A number of people have asked for details of our email addresses. The possibilities, in decreasing probability of getting here, are: on CIX as dotexe@cix (we visit at least once a week), on Compuserve 100014,2407 (we browse), and via Internet as dotexe@cix.com.uk (we'll be there, but watch for bouncing mail).

New Periscopes

Periscope/EM is a new addition to the Periscope debugger menagerie. It's a software version of the low-end Periscope I hardware debugger, using 386MAX's VCPI interface to hide and write protect itself up in extended memory. It needs a 386, a copy of 386MAX or BlueMAX V5.11+, around 300 KB of memory past 1 meg, and a 32 KB workspace in DOS RAM. The UK price is £155. 386Max or BlueMax costs extra - £70 or £90.

The full ICE in the Periscope range, the Periscope Model IV, has also been revamped, with a new motherboard and pod to handle 486 pin-outs. The new Rev 2 board can work with faster processors - up to 33 MHz - and has better tracing and a 4 KB buffer expandable to 16 KB. Periscope IV Rev 2 costs £1450 with 4 KB, £1745 for 16 KB. The 486 pod can be bought for £395. Periscope is distributed by Roundhill. Telephone 0672 84535.

Flowchart Kiev

Technosoft is a Kiev-based company (now there's an opening phrase we haven't used before) which offers a flowcharting system called R-Tech. The program runs under MS-DOS in conjunction with either Turbo C or Turbo Pascal. Using a special (text mode) Interactive Development Environment, you write your program using flowchart conventions to control program flow, and either C or Pascal for the statements. When you have finished, you 'compile' your diagram to produce ordinary C/Pascal, and the Borland compiler is invoked to generate a .EXE file.

R-Tech is unusually complete and smooth for this type of product - according to the manual's blurb, it has been under development for over 20 years. You can also buy a source level debugger (which operates in a similar manner to Turbo Debugger) and a reverse engineering module, which converts existing C/Pascal code into its flowchart equivalent. The basic package costs £360 (for just one language), the add-ons are a further £120 each. UK contact is VRBA & Associates: 0983 611119.

Testing NLMs

Nu-Mega, who .EXE readers will know as the maker of the PC memory protection utilities Soft-ICE and Bounds Checker, has released two new utilities for Netware 386 developers. The first has a broad parallel with its DOS products - it's a memory protector for Novell servers. NetWare 386, like DOS, runs without using the 386's memory management features, so any NetWare Loadable Module (NLM) can conceivably poke anywhere in memory. Nu-Mega's NET-Check runs all the NLMs in virtual mode, kicks in the 386 MMU, and automatically write protects code areas, address space above real memory and the first 4 KB of RAM.. The second program is NLM-Profile, a utility for monitoring profiling NLMs' processor use. Handy, given NetWare's non-preemptive scheduler and the greediness of some modules.

NET-Check V1.0 sells for $499; NET-Profile costs $199. Nu-Mega can be contacted on 0101 603 888 2386.

Gupta/Novell Tie-In

Novell bought a 20% stake in Gupta Technologies this April : the first fall-out products from the buy-out are now arriving. SQLWindows for Btrieve is, as you'd imagine, Gupta's Windows application development product linked to Btrieve, the database bundled free with every copy of Novell NetWare. The product currently enables applications to link with Gupta's own SQLBase, and, via SQLNetwork, to remote DBMSs such as DB2 or Oracle. SQLink for Btrieve will be marketed by both companies, and is available for £1,595. The package includes the standard SQLWindows Development Kit, five runtime versions of SQLWindows and the Btrieve library. Gupta Technologies is on 071 333 1417.

Turbo Debugger

The Turbo Debugger package from Borland has a remote option in which a small 386 Ring 0 application runs in a client machine and reports the machine state - via the serial port - to a Turbo Debugger front-end on another PC. All the TD options - breakpoints, single stepping and so on - are supported, and the snazzy interface remains the same. Now, a company specialising in embedded systems software has come up with a background monitor for 8086-compatible embedded systems that produces the same serial infomation as the remote Turbo Debugger code. Thus, any 8086-compatible system can now be debugged with TD. The driver rakes up around 5 KB of ROM and 2 KB of RAM in the client ; full source code is provided. Serial drivers for the 8250, 8251, 8256 and NEC V25/35 and V40/50 are also included. The TDREM package costs £194. It requires another utility called LOCATE, which provides embedded start-up and run-time libraries for Microsoft C, Turbo C++ and Turbo C code. LOCATE costs £304, and both are available from Great Western Instruments on 0761 52116. LOCATE and TDREM are products of Paradigm (*sic*) Systems.

digital

Workstations from Digital.

To get the most out of your applications ...

CIRCLE NO. 745

Letters

We welcome short letters on any subject that is relevant to software development. Please write to The Editor, .EXE Magazine, 10 Barley Mow Passage, Chiswick, London W4 4PH. Unless your letter is marked 'Not for Publication', it will be considered for inclusion on this page.

May appeal

Sir,

I don't understand your concern. If you want to receive more letters, let Jules (Mr One Bug) May write a regular column.

The Verity Stob T-Shirts look nice.

Chris Brooks
PC Analyst Programmer
Southdown Building Society
Lewes
East Sussex

Good idea! You will find a teaser for Jules' column in this month - the real thing should start next month. Give the man a T-shirt.

TEX support

Sir,

Dan O'Brien's two-column listing routine was very interesting, but there are also other solutions: my own uses the TEX typesetting language:

```
\input twocol.style
\input verbatim.style
\begindoublecolumns
\fontstyle elite,6pt
\askfor\sourcefile
\verbatim\sourcefile
\finish
```

This has the advantage of being short, wholly machine and device independent and will wrap-around overlong lines (with indentation) and balance up the last page into two equal-height columns, leaving space for comments underneath.

You remain the best systems development magazine on the market: keep up the good work!

Peter Flynn
Cork
Ireland

Neural Hype

Sir,

I was very interested to read the article about Artificial Intelligence by Darrel Ince in your March issue. I think it is important to keep the recent developments in perspective. There is a great danger of this technology being 'hyped', in a way that was so detrimental to expert systems technology 10 years ago.

Our research has shown that the multi-layer perceptron model of neutral networks, which I believe is the most commonly used, is a form of non-linear estimation. In this sense it belongs to a class of techniques that was already in use in complex statistical analysis. Indeed, there is reference to these in the article about the NAG FORTRAN libraries, in the same issue.

Why then are neural networks creating so much excitement? The answer is usability. This approach encourages a frame of mind that is not hampered by the preconceptions of statistical approaches, and people have applied the technique to data sets that a statistician would never have considered worthy of attention. Further, these techniques have been packaged in a form that is readily usable by non-mathematicians. This is both its strength and its weakness. One sees the most outrageous claims made about the power of the technology.

Are neural networks revolutionary? I think they represent a revolution not of techniques, but in applying those techniques. Our view is that they are a useful tool, along with, not instead of, our existing tools for expert systems, natural language processing, and symbolic computing; and like these tools, in order to be fully exploited they must be tightly integrated within traditional computing environments. We will continue to use the tool best suited to the specific problem presented by a client.

Ted Walker
Expert Systems Ltd
Oxford

More bl**dy paradigms

Sir,

So a paradigm is just a 'pattern or example' (.EXE, October '90, p10), is it? I suggest that, when properly used, it means rather more.

A paradigm is an accepted model or pattern which underlies 'normal science', that is, research based upon past scientific achievements 'that some particular scientific community acknowledges for a time as supplying the foundation for its further practice'.

Normal science attempts to bring theory and fact into closer agreement, but there are always some discrepancies. Normal science typically perceives such discrepancies as 'puzzles', but they may also be viewed as anomalies in the paradigm. A 'crisis' occurs when a discrepancy comes to be viewed as more than 'just another puzzle' or normal science. The recognition that an existing paradigm is inadequate leads to a scientific revolution, in which the older paradigm is replaced in whole or in part by an incompatible new one (see *The structure of scientific revolutions*, T S Kuhn, 1962).

A paradigm has been variously defined as a 'strong network of commitments, conceptual, theoretical, instrumental and methodological, the source of the methods, problem-field, and standards of solution accepted by any mature scientific community at any given time', and 'universally recognisable scientific achievements that for a time provide model problems and solutions to a community of practitioners'.

Whither 'example'?!?

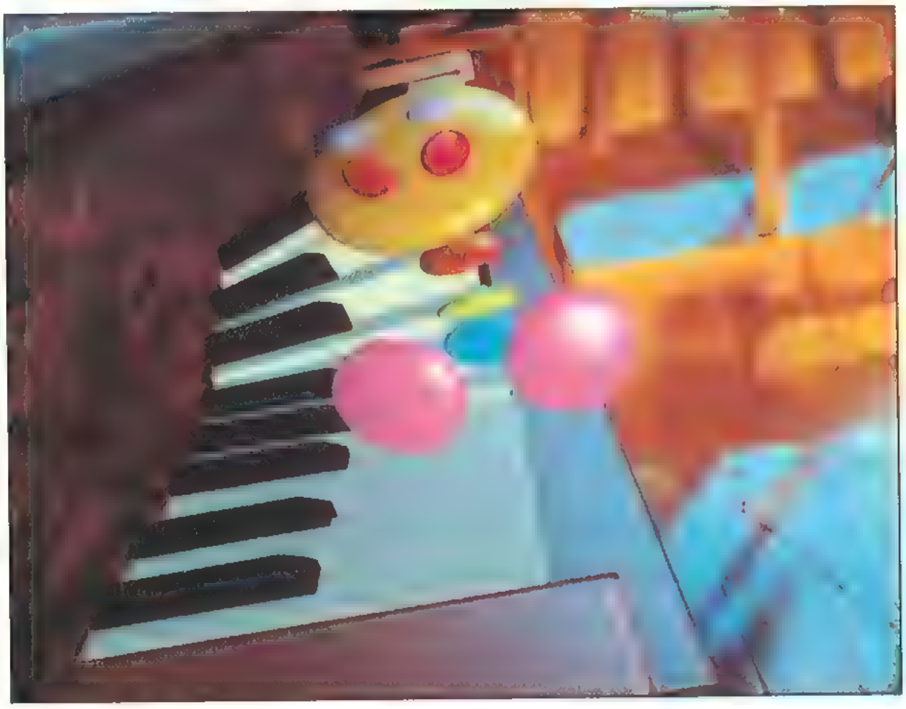
Simon Ashworth
London SW11

Letters submitted to this page may be edited. The writer of the best letter of the month, as judged by the Editor, will be rewarded by a T-shirt or similar-valued .EXE trinket. The best letter is the one printed first.

Smartie People

Animated computer graphics are very fashionable in television commercials. But how are they made? Who makes them? Al Roth explains.

You have no doubt seen in TV adverts an increasing tendency to use computer generated graphics and animation. The Smarties advert, in which dozens of the little cuties spew out of a saxophone and spin merrily around a computer-generated room, is a classic example. The technology underlying this trend is severely powerful. The hardware and software facilities needed are well beyond that available to the average .EXE reader, but we thought it would be fun to take a look at what is being done, and give one or two hints for the more ambitious. If you can get your PC to do this, then you are verily a non quiche-eating programmer.



Background

When a television or film company needs a computer animation sequence, it usually approaches a specialist graphics organisation. The customer usually has some idea how the sequence should look, how long it should last, and a notion of the message to be passed across to the unsuspecting public. Many computer graphics companies require that the character and sequence is drawn beforehand by the customer - that is, the sequence entirely animated by hand, and then given to the graphics company to replicate within a computer animation. Others believe that this is unnecessarily restrictive, preferring instead to design the whole sequence interactively on the machine - avoiding the limitations of being tied to a paper sketch.

The main stages in creating an animated sequence are model building, animation and rendering. Building the model may involve creating a company's logo, or a Smartie, or a saxophone. Even with specialist model-building tools, this can take time (eg more than a day to build the Smarties piano). Animation refers to the process of scripting the motion of objects and the camera. The final stage is rendering the wire-frame characters so that they appear as finished. Rendering is probably the most time consuming and computationally expensive stage, taking between 5 and 50 minutes for each frame. As we shall see, each stage requires a different set of skills and support technology.

Smarties

The Smarties adverts for Rowntrees were generated by a company called Snapper Bytes using a Symbolics LISP machine. Object-oriented programming played a vital role in the Smarties software written by Bruce Steele and directed by Matt Forest of Snapper. OOP makes the task much easier for the programmer, especially at the modelling and animation stages. Define a `Smartie` as a class, give it behaviour in the form of methods, send messages telling it what you want it to do. Then create numerous instances of the class 'Smartie' and off you go!

The Smarties sequence was created using the LISP programming language. Associated with every LISP symbol there is a tabular data structure called the 'property list' (or *plist*). The plist makes it possible to give a symbol a set of attributes, each of which has a given value. Put simply, a plist is a place where information can be stored. When a symbol is first created its plist is empty. Properties are assigned using the LISP primitives `GET` and `SETF`. Figure 1 shows some noddy LISP code which uses these primitives and stores some attributes about a symbol called MAN. The attributes in the example are `NAME`, `SIZE` and HEIGHT although it would be very easy to store any kind of property we would never want to associate with a MAN.

Property lists are a fast and efficient mechanism for attributing values to key characteristics associated with a symbol. Note that the same effect could be provided using the

Common LISP Object System (CLOS). This is the object-oriented standard for programming in LISP, and allows values to be stored in *slots*. However, plists are still useful because, unlike slots, they are dynamic: - you can stuff another value into a plist without changing the class definition or structure.

For the Smarties advert, the animators made extensive use of plists. The attributes of the Smartie 'skeleton' are stored on the plist of the top-level object. That is, having defined a higher-level object (called an *actor*), Steele used the plist to stash information determining the body parts to associate with that particular skeleton. Plists are also used by the S-Geometry package which models the Smarties. In this case, the object's plist contains all the attributes (surface colour, texture maps etc) that will be needed to render the object later on.

```
;;; Property Lists can be accessed by the primitive GET,
;;; values assigned with SETF

;;; So if we had a symbol called MAN and we had properties
;;; called NAME and SIZE we could retrieve their values
;;; like this:

(GET MAN 'NAME)
                AL

(GET MAN 'SIZE)
                BIG

;;; The LISP form SETF may be used with GET to add a new
;;; property, or to change the value of an existing
;;; property:

(SETF (GET MAN 'HEIGHT) 'TALL)
                TALL

(SETF (GET MAN 'SIZE) 'HUGE)

;;; the last statement increases the SIZE property
;;; associated with the symbol MAN.
;;; Typing (GET MAN 'SIZE) will now return HUGE
;;; instead of BIG.
```

Figure 1 - Property Lists

Cheating Physics

An important aspect to the work of a computer animator is to generate images which are pleasing to the eye. Sometimes this means 'bending' reality to fit with the image the artist is trying to create. The Nintendo advert, for instance, contains a sequence where the 'camera' is zoomed out from a shot of a human face to a globe full of people taken from 22,000 miles above the Earth. This is mathematically impossible - but it is still believable.

Cheating physics is not always so easy. One animation (used for a point-of-sales promotional video for Boots) contained a robot which was to be visible from the waist upwards. The model contained seven different rotation axes for each arm. In order to make it seem believable in the tracking shot used in the final sequence, the animators were obliged to observe the character from several different camera angles. Having tweaked the robot so that it looked fine viewed head on, they would discover that it appeared grotesque when viewed from the side. As Bruce Steel admitted, 'It was only too easy to make him look like Joe Cocker at Woodstock.'

When viewed from different angles, an object appears to undergo some displacement. This apparent law of Physics is consistently cheated by animators to produce a visually satisfying effect. So when drawing characters from different angles, an animator will often alter the extent and direction of movements of arms and legs.

The McEwans Low Alcohol advert 'Walk in a Straight Line', which concerns the adventures of a capital letter 'A', depends entirely on displacements. Steele points out that with all that twisting and stretching it would have been very easy to make the 'A' look as though it had been in a road accident. Furthermore, the combination of several displacements working on bits of the same object was difficult to predict, necessitating a high degree of trial and error. Steele notes, 'Moving the point at which a displacement starts by a couple of frames could make the difference between looking cool and looking drunk. It's important to get this right when the product you are trying to sell is low alcohol lager.'

Invisible Wires

The eight Smarties were moved using *trajectories*, which act a bit like wires in mid-air. When the Smarties all fly together to form the body of a Smartieman, the 'skeleton' of the creature is a set of these wire-trajectories. Manipulation of the Smartie character was achieved by manipulating these trajectories. A similar technique was also used in the McEwans advert, where the A character is required to dive into a pint of beer, generating hundreds of bubbles. These bubbles were arranged along a set of wire trajectories, with 100 or more on each trajectory. From then on it was only necessary to think about what the three trajectories were doing - the animators could forget about all the individual bubbles.

For another animation, Bruce Steele generated a wire-frame character of a skeleton that knows how far its elbow can bend

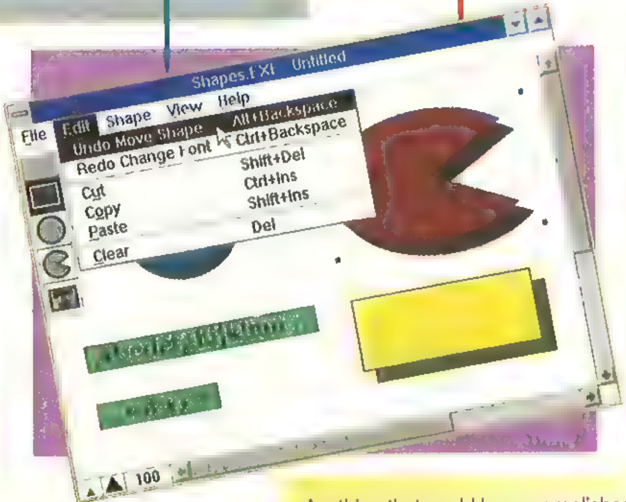
Limehouse's Low Alcohol 'A'

In the example programs you can see how *the GUI_MASTER* takes care of all the standard CUA functions like

- Scrolling
- Runtime error handling
- Printing
- Undoing/Redoing the most recent user actions
- Text editing
- Standard mouse interactions such as dragging and sketching
- Archiving objects
- Standard dialogs such as open, save as, fonts, color picker
- Using the clipboard

The 7 example programs range from simple to fairly complex. Together with the cookbook we provide they help you gain insight in all the possibilities of the *GUI_MASTER (Class Tree for C++)*. They come complete with fully annotated source code, so you can even use them as a basis for your own applications.

Effectivity MS Windows Presentation



The *GUI_MASTER (Class Tree for C++)* enables you to specify different worlds, different representations of the same data which can be shown in different parts of the same window. If you modify some of the data in one world this is automatically reflected in the other world.

If you use *GUI_MASTER* your application is, as a matter of course, structured around the Data/World concept. An excellent basis for GUI applications.

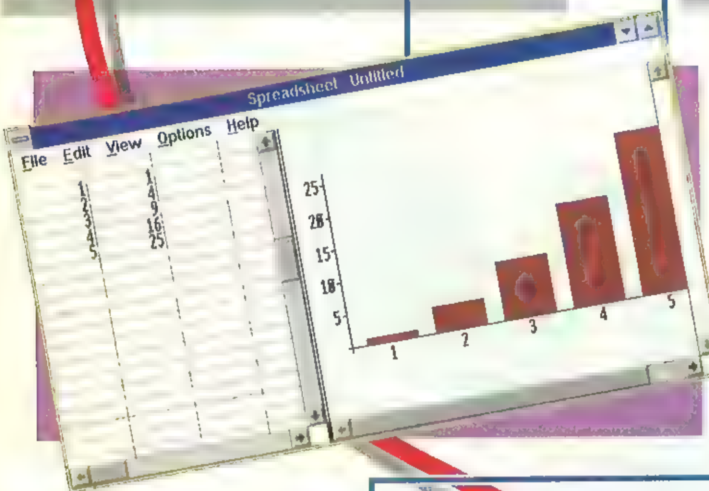
Anything that could be accomplished through 'normal' programming can be done if you use *GUI_MASTER*. It's just that *GUI_MASTER* makes it all a lot easier. Many of the things you might not include in your application because you think that they're too complex or time-consuming to program, are easy to make with *GUI_MASTER*.

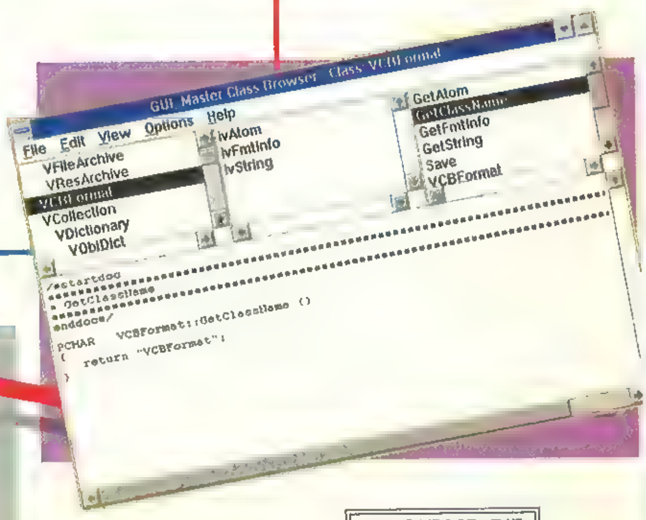
If you are building sophisticated C++ applications why not benefit from our *GUI_MASTER (Class Tree for C++)* to give you

- a head start in building GUI applications
- a GUI toolkit consisting of 85 reusable classes, together forming a well-tested and fully functional 'standard application'
- platform independence between MS/Windows and OS/2 Presentation Manager

The product offers

- 7 example programs
- a Class Tree
- a C++ Source Browser





The class browser is of crucial importance to enable you to view programs as collections of classes and not as a set of source files. It enables you to examine code made by others, and to understand the class structure of that code, thereby promoting the concept of 'the programmer as reader' and stimulating the reuse of software.

With our browser you can

- browse through the source code of any C++ program
- launch the editor of your choice to modify your C++ programs
- quickly locate any method in the Class Tree

CIRCLE NO. 747

before it locks, and how to put its feet to the ground. When the character was moved, the body segments automatically position themselves on appropriate parts of the skeleton. Steele explains 'You can then animate the stick figure and the rest takes care of itself. If the skeleton was written in C, it would be necessary to replicate code explicitly for each skeleton.'

An Object-Oriented Approach

The conventional approach to designing 3-D interfaces has usually been task-oriented, like the languages in which they have been implemented. If you wanted to rotate an object around an axis, you would write a function called `ROTATE` and pass it the name of the object to be rotated, the axis and the amount of rotation. In an object-oriented interface, you send a message to the object saying `ROTATE` and pass it the axis-name and the amount of rotation. The object would find the axis and rotate itself by the requested amount. This is illustrated in Figure 2.

The difference between the two approaches seems small in this example. The power of OOP only becomes apparent when there are numerous types of objects - say points, edges, faces, polygons etc. In the conventional approach you would encode individual functions called `ROTATE-POINT`, `ROTATE-EDGE` and so on. In an object-oriented system it is only necessary to send the object one message - `ROTATE`. The programmer does not have to remember which function to call; instead the object receiving the message knows what to do with it.

Moving the displacement by a couple of frames can make the difference between looking cool and looking drunk

Another advantage is that once you have described the behaviour of a given construct (say a Smartie), then the same behaviour maps well onto any other Smarties that you create. Conventionally, if the animator had to generate a sequence of three robots moving along a road, he would have to create a function which ensured that they did not collide. If the number of robots was subsequently doubled, the whole function would have to be rewritten to cope with six robots. Using object-oriented technology, each object is responsible for itself. You can define the problem for one robot and then add as many copies as you like.

Bruce sums it up by saying: 'Its really a balance between letting go of the reins and

The Hardware and Software Vendors

Computer animation soaks up a lot of computer power. Accordingly, there are a number of specialist machines and software packages geared to supporting graphics programming and computer generated animation. Two major players in this market are Symbolics and Thomson Digital Image.

The Symbolics system was used by Snapper Bytes to generate the Smarties commercial. Commenting on this choice, Bruce Steele said 'Symbolics machines may not be not ideal for rendering, or for all of 3-D graphics, but they make model building, animation and rendering far more interactive.'

Symbolics offer a suite of software packages that are aimed at supporting the various stages. S-Geometry supports the modelling phase by providing a range of geometric and topological tools allowing the user to interactively edit an object and manipulate points, segments, faces, vertices, edges and polygons. It is also possible to position lights and maps and attach them to the model, as well as manipulate the camera.

For scripting and choreographing the animation, the S-Dynamics package provides tools for transforming the position, shape, map coördinates and render attributes of objects. The tools displacement feature allows the animator to control the way an object's shape changes over time helping to achieve an effect of fluid motion. S-Dynamics also allows for the definition of trajectories - visual references about the path of an object through a complex environment.

The user can tune a trajectory with respect to other elements, so that when a trajectory has been defined it is possible to move the camera or any number of objects along it.

S-Render provides support for the rendering process. It makes it possible to add colour and texture to the 3-D models and enhance each frame using advanced lighting techniques, or by selecting different surface properties for individual objects.

An alternative approach is offered by the TDI organisation, which was formed in 1984 by three engineers from the simulator division of Thomson CSF. After joining forces with the Computer Graphics Laboratory and the subsequent acquisition of Sogitec, France's leading producer of award-winning 3-D animation, TDI became a major force in 3-D computers design and animation software for the television and film industry. In 1989 IBM France purchased a 49% share in the company and TDI established a North American subsidiary. TDI now has more than 60 staff, with more than 350 Explore systems in use worldwide.

TDI Explore is an interactive 3-D modelling and animation system running on the entire range of Silicon Graphics workstations. TDI Explore has a number of modules for modelling, animation, material editing, image editing and output to video devices.

The modelling suite contains an interactive 3-D polygon modeller which allows a variety of operations on polygons and a second module for generating 3-D curved surfaces and interactively manipulating lines, profiles, networks and surfaces.

The animation software allows the user to control the motion of references, light sources, the virtual camera and manipulate variables such as timing, path, trajectories and the characteristics of light sources. Additional modules make it possible to animate articulated structures, and allow users to check their animation with real-time or manual controls before rendering the images.

```
;;; This gives you the object-oriented code for building
;;; a new flavor called bone on the Symbolics machine.
;;; (Common LISP uses classes instead of flavors.)


(DEFFLAVOR BONE ()
     (3D:OBJECT)
     :INITABLE-INSTANCE-VARIABLES
     :READABLE-INSTANCE-VARIABLES
     :WRITEABLE-INSTANCE-VARIABLES)


;;; The above creates a class definition for BONE and
;;; also says that it is to inherit from another class
;;; called 3D:OBJECT. It will do everything that a
;;; 3D:object does.


;;; To convert an existing 3D object into a bone we
;;; define a method. Note that :PUTPROP serves the same
;;; purpose as SETF (GET symbol property) value. This is
;;; due to the Symbolics-specific syntax (and the way
;;; plists are implemented on the LISP machine).


(DEFMETHOD (:CHANGE-INTO-BONE OBJECT) ()
     (CHANGE-INSTANCE-FLAVOR SELF 'BONE)
     (SEND SELF :PUTPROP '(T) 'BONE)
     (SEND SELF :PUTPROP 0.0 'CUR-ROT)
     (SEND SELF :PUTPROP -20.0 'MIN-ROT)
     (SEND SELF :PUTPROP 20.0 'MAX-ROT))

;;; The method below rotates a bone and imposes some
;;; constraints on how far it can be rotated. Note
;;; that the :%ROTATE BONE method is a specialisation
;;; of another method defined for 3D object
;;; :%ROTATE OBJECT.  Previous to this, the BONE
;;; would use the rotate method defined for 3D:OBJECT
;;; instead of its own more specialised method.


(DEFMETHOD (:%ROTATE BONE) (DEGREES AXIS-VECTOR CENTER)
     (LET ((MAX (SEND SELF :GET 'MAX-ROT))
           (MIN (SEND SELF :GET 'MIN-ROT))
           (CUR (SEND SELF :GET 'CUR-ROT)))
     (IF (PLUSP DEGREES)
          (IF  (< (+ DEGREES CUR) MAX)
               (SETQ CUR (+ CUR DEGREES))
               (SETQ DEGREES (- MAX CUR))
               (SETQ CUR MAX))
          (IF  (> (+ DEGREES CUR) MIN)
               (SETQ CUR (+ CUR DEGREES))
               (SETQ DEGREES (- MIN CUR))
               (SETQ CUR MIN)))
     (SEND SELF :PUTPROP CUR 'CUR-ROT)
     (ALTER-ALIGNED-MATRIX BASE-MATRIX
          AXIS-VECTOR CENTER :Z-ROT DEGREES)))


;;; Note that the minimum, maximum and current rotation
;;; are all stored on the property list of the object.
;;; They could have been stored as instance variables.
```

Figure 2 - Defining a bone object

letting the computer take over within the constraints that you have set, and actually keeping control of the bits that you want'.

A C Approach

Not everyone is using object-oriented techniques. UK company Electric Image specialises in producing computer animation sequences. The company uses its own proprietary software in addition to software and hardware provided by French organisation Thomson Digital Image (TDI), and is responsible for a large number of the TV adverts that you will have seen. One of its early efforts, a very small 3-D graphics system known as the CBG2, is still used to provide the graphics for the ITV Chart Show.

The company has recently developed a rendering program called Synthacam. Implemented on the AT&T Pixel Machine Supercomputer (a parallel machine), the Synthacam system allows the generation of images displaying motion blur, depth of field, refraction, reflection, and other types of real world light interaction. The company claims that, prior to Synthacam, such imagery required render

A fast car produced with the Symbolics System

times of 30 to 40 hours per frame, but can now be realised in minutes.

Electric Image Technical Director Stuart McEwan said: 'We don't use object-oriented technology (C++) because we don't have it available yet. When you are writing rendering software you are using the bare bones of C. This is because you are more concerned with the actual machine control and the optimisation aspects. The thing about OOP is that it is good for the programmer, but it doesn't necessarily produce very efficient code.'

Electric Image decided that LISP was insufficiently portable for its kind of work, placing too much reliance on a particular operating system. The company has five different types of computers, and its code has been ported to all of them. McEwan said, 'If we had written our code in LISP then it would have been impossible.'

McEwan believes that, for the modelling stage, object-oriented technology can provide a better environment for the programmer, but the trade-off is low-level efficiency. He explains 'When you are generating code to do the rendering you are interested in speed. When you are doing modelling, on the other hand, the most important thing is the efficiency of the animator himself.'

Dozens of the little cuties spew out of a saxophone and spin merrily around a computer-generated room

The Future

Whatever the technical differences between animators, they all agree on the fundamental issues. Computer graphics is still in its infancy. According to McEwan 'True photorealism won't be realised for at least five to 10 years, because the right technology doesn't yet exist. We still need another order of magnitude of rendering capability or computing power to make it happen.'

Bruce Steele anticipates some very interesting technical developments. The introduction of tools based on virtual reality will change the way his business is done, by allowing the animator to get in and move around within the animation. The animator of the future will be able to view all the components, their relative positions, and camera as though he was a participant in the scene. EXE

Al Roth is a freelance writer, technology consultant, and part-time structural support for the M62 fly-over. Hobbies include eating, going on long holidays, and smiling smugly at FORTRAN programmers. He can be contacted at alroth@cix.

The author would like to thank Bruce Steele of Limehouse, and Stuart McEwan of Electric Image for their invaluable comments and assistance in the preparation of this article. Many thanks also to Symbolics Ltd and Thomson Digital Image.

Smarties are available from your local sweetie shop priced around 20p for a tube of around 40 sweets.

3D Computer Graphics - The Basic Transformations

3D computer graphics is fascinating - but the programming is hard work. In this new series of articles, Graeme Webster will present working code which you can build up into your own 3D graphics programs.

Computer graphics is one of the most rewarding of programming activities. It is also one of the most challenging. This is particularly true of 3D work for which standards are still in a state of flux and comprehensive subroutine libraries and tool-kits not readily available. Building even a modest piece of 3D software requires a lot of code. User interfaces are especially difficult and there is a real danger of falling into the trap of giving the users what we know how to implement rather than developing what they really need in order to solve their problems. For the most part, this is because the 3D world is much richer than the 2D. It took many years and false starts to get the user interface for 2D graphics and data manipulation about right. 3D is intrinsically much more difficult. There are mechanical problems such as the lack of any effective, available 'off the shelf' 3D pointing devices equivalent to mice and digitising tablets for 2D. Much more fundamentally, there are problems related to human psychology and the perception of the third dimension (see separate box).

There is another problem. Most of the published material on 3D graphics is either in research papers or in text books which justify their advanced status by only giving the algebra or perhaps an outline sketch of a program in pseudo code. Rarely do they publish a piece of working code which you could just copy down and use. (A bibliography of Dr Webster's recommended books is available by writing to the Editorial office - Ed).

The principle difficulty with formal books is that what they give is all very fine but it is often a very long way from an implementation. There are a few notable exceptions to this. In his books *Programming Principles in Computer Graphics* and *Interactive 3D Computer Graphics*, L Ammeraal gives the C code for a 3D line drawing package. *Fundamentals of Three-dimensional Computer Graphics* (by A Watts) covers more topics and gives code for a *z*-buffer renderer (we will come to this in a later article). I very much welcome this approach. This series of articles will try to follow it by giving working subroutines that can be used together with your own, non-graphics, code to do 3D graphics.

Transformations

There is no royal road to 3D graphics and ultimately we have to get down to some algebra. The objects with which we will be dealing are points in 3D space, lines joining

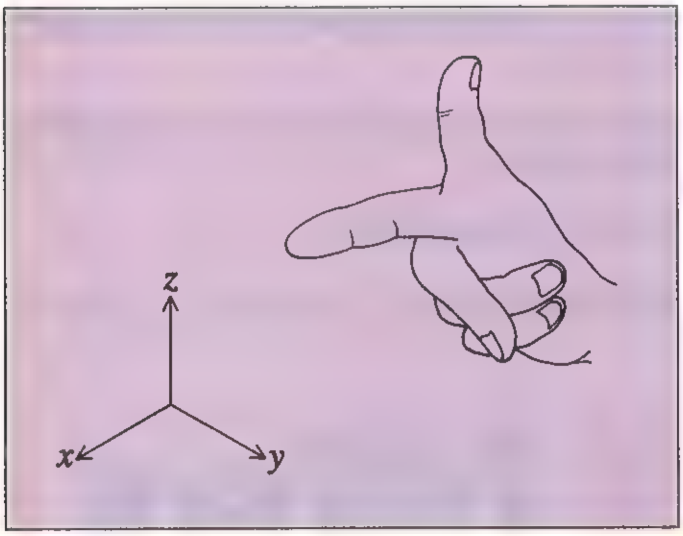


Figure 1 - Testing for right-handedness

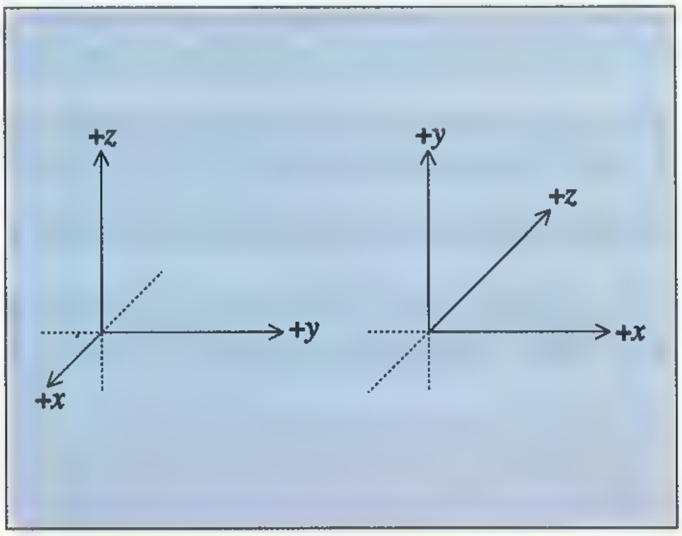


Figure 2 - Conventional Systems of Axes

NDT

EXCLUSIVE

80386-40

Landmark (vr.99) 70Mhz

SYSTEMS ARE MANUFACTURED WITHIN THE UK USING AMERICAN COMPONENTS EXCLUDING THE CASE/POWER SUPPLY, KEYBOARD AND FLOPPY DRIVES

AMERICAN COMPONENTS

80386-40 AMD CPU 64K Cache
2 Mb 70ns SIMMS Memory
Mini Tower and Power Supply
2 Serial and 1 Parallel Ports
Choice of floppy drive
Super VGA Graphics Card
14" 8514a Colour Monitor 1024 x 768
110Mb 9ms Hard Drive 64K Cache
Keyboard and Mouse

2 YEARS ON-SITE MAINTENANCE

£1595*

* ALL SYSTEMS COME CONFIGURED AND FORMATTED READY TO GO. INCLUDING MSDOS 4.01 DELIVERY £25 (INSURED) - ALL TRADEMARKS RECOGNISED

UPGRADES & ADD-ONS

210Mb 9ms HDD	£300	120Mb Internal Tape Unit	£190
380Mb 16ms	£820	250Mb Internal Tape Unit	£99
670Mb 15ms	£1100	External Drive Kit	£99
1,2GB 15ms	£1550	Additional Tape Controllers	£50
80486-33 CPU ISA	£800/£1100	Additional Floppy Drive	£50
256K Cache	£250	IIT Maths Co-Processor	£320
HDD Doubler	£175	Extended Warranty to 3 years	7%
Cache Ctrlr (0.5ms) 512K	£350		
1Mb Video Card (+512K)	£50		
Per 2Mb SIMMS	£80		

ADDITIONAL COSTS

NEW DIMENSION TECHNOLOGIES
18 CORPORATION ST WEST
WALSALL - WEST MIDLANDS
WS1 4DU - ENGLAND

Phone! 0922 616733
Fax! 0922 616737

CIRCLE NO. 752

points, polygons defined by a set of points all in the same plane (which can be in any orientation) and solids bounded by a set of polygons. The definition of a point in 2D by *x*- and *y*-coördinates, taken with respect to two mutually perpendicular axes, is pretty familiar. In 3D the concept of rectangular Cartesian Coördinates, to give them their proper name, is extended to have *x*-, *y*- and *z*-coördinates. Unfortunately, there are two distinct ways of choosing the *z*-direction. If *x* and *y* lie in a horizontal plane, *z* can be either up or down. We will use the convention of having *z* up. Such a coördinate system is said to be right-handed. You can test handedness by using your right hand. Extend the index finger and then bend the second finger to be at right-angles to the palm and hold your thumb upward at right-angles to both the index and second finger. Point your index finger along the positive going direction of the *x*-axis and your second finger along the positive direction of the *y*-axis. If the axes are right-handed your thumb can be aligned with the positive direction of the *z*-axis as in Figure 1. If not, the axes are left-handed.

The orientation of the axes in space is also arbitrary. I was brought up in the European mathematical tradition which has the *x*- and *y*-axis horizontal, as though they were drawn on a piece of paper lying on a desk. The *z*-axis then sticks vertically upward as in Figure 2. This is the convention used by many architectural graphics systems. It is also the one used by Ammeraal in his books and is adopted here. The influence from America and many engineering CAD systems is to think of the display screen as though it were the surface of an upright drawing board. The *x*- and *y*-axes are then drawn horizontally and vertically on a vertical CRT screen with the *z*-axis pointing away into the screen, Figure 2. These axes are then left-handed.

The set of coördinates used to define objects are called world-coördinates. They give objects absolute location in what is referred to as object-space. Each point requires three, generally floating point, numbers to specify its location. These are intimately bound together so it usually makes good programming sense to define a structured variable to hold them, such as

```
struct Vertex3Struct
{ float x, y, z;
}
```

A key task is to transform these into the screen-coördinates X,Y of corresponding points on the screen, referred to as image-space.

```
struct Vertex2Struct
{ float X, Y;
}
```

Conceptually, this transformation is equivalent to taking a photograph, the camera converting points in the 3D world into equi-

The Human Factor

The eye-brain combination is a wonderful image processing and analysing system. It was, however, evolved for hunting and survival in the hostile world of our prehistoric ancestors and not for 3D visualisation. It may, in fact, be best at trying to find 2D patterns in fragmented images, a tiger partially obscured by a tree, for example. This could explain why some people see faces in clouds or why the ancients divided up the stars into pictorial constellations. It has been suggested that the earliest painters at Lascaux did not consciously set out to draw particular animals but enhanced only what they perceived already to be there in the shadows cast by the bumps on the cave walls. It is also a curious fact that though the 3D world is richer in shapes than the 2D, we have a comparatively limited vocabulary to describe them. This points to a lack of perception or at least to a lack of importance of 3D description in evolution (when something is important there is usually a rich vocabulary to describe subtle nuances of it, for example the Eskimo people's many words describing the different types of snow). Nevertheless, as Escher observes:

'Our three-dimensional space is the only true reality we know. The two-dimensional is every bit as fictitious as the four-dimensional, for nothing is flat, not even the most finely polished mirror. And yet we stick to the convention that a wall or a piece of paper is flat, and curiously enough, we still go on, as we have done since time immemorial, producing illustrations of space on just such plane surfaces as these. Surely it is a bit absurd to draw a few lines and then claim: "This is a house".'

Despite this, we still want to make pictures on our computer screens which look convincingly enough like houses.

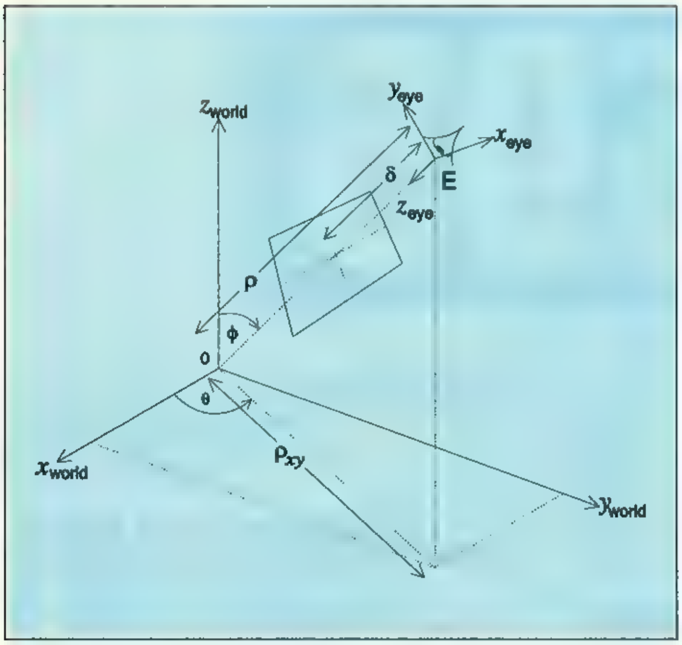


Figure 3 - World- to Eye-coördinate Transformation

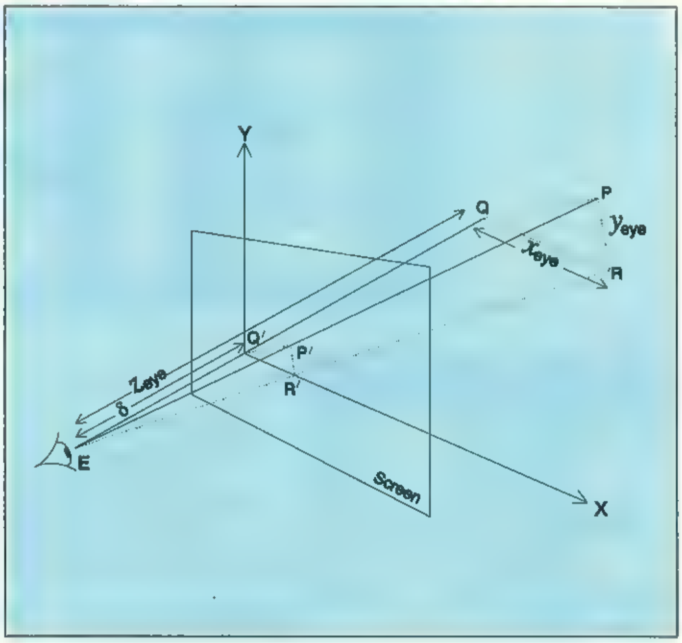


Figure 4 - Perspective Transformation


Microsoft C
Microsoft Windows
Software Development Kit
Microsoft
WINGZ
Ami Ami
DA VINCI eMAIL
Microsoft Project for Windows
Microsoft Word for Windows
COREL DRAW!
Designer
Aldus PageMaker

valents on the 2D film. The exact transformation depends on the location of the camera or eye, the spot at which it is pointed and the relationship of the film size to the focal length of the lens being used. It is helpful to break this process down into two stages. The first is a general transformation of the world coördinates into a new set of 3D coördinates based on the location of the camera (these are usually referred to as eye-space coördinates). The second transformation is more display device specific taking the eye-space coördinates to, for example, screen coördinates. Symbolically:

$$\text{World-coördinates}(x_{world}, y_{world}, z_{world})$$
$$\downarrow$$
$$\text{Eye-coördinates}(x_{eye}, y_{eye}, z_{eye})$$
$$\downarrow$$
$$\text{Screen-coördinates}(X, Y)$$

The computationally more expensive world-to-eye transformation is generally needed less frequently than the eye-to-screen projection, so that by separating them it may be possible to speed execution. In passing, a minor irritation is that usually we will want the screen-coördinates to have their origin in the centre, with the X-axis running horizontally to the right and the Y-axis vertically upwards. Most screen graphics systems are conditioned by

```
#include<ctype.h>
#include<graph.h>
#include<math.h>
#include<stdio.h>
#include<string.h>
#include <video256.h>

#define TRUE 1
#define FALSE 0
#define MAXVERTEX 1024
#define MAXEDGE 1024
#define PiOn180 0.0174533

struct Vertex3struct
{ float x,y,z;
} Vertex3[MAXVERTEX];

struct Edge3struct
{ short start,finish;
} Edge3[MAXEDGE];

unsigned short NumVertices,NumEdges,HRes,VRes,
              LineCol=255,TextCol=243,BkgdCol=0;
unsigned char Cmd[3],CmdTail[81];
float GazeX=0.0,GazeY=0.0,GazeZ=0.0,
     EyeX=300.0,EyeY=0.0,EyeZ=0.0,AngView=45.0,
     WE11,WE12,WE13,WE21,WE22,WE23,WE32,WE33,WE43,
     ViewDist,
     MoveX=0.0,MoveY=0.0,MoveZ=0.0,
     CentreX=0.0,CentreY=0.0,CentreZ=0.0,
     RotateX=0.0,RotateY=0.0,RotateZ=0.0,
     ScaleX=1.0,ScaleY=1.0,ScaleZ=1.0;

void CalCoeffs(void);
void CubeData(void);
void DoPerspective(float x, float y, float z,
                   short *scnx, short *scny);
void DrawView(void);
void GetCentreOfGaze(void);
void GetCommand(void);
void GetData(void);
void GetEyePoint(void);
void Move(void);
void Quit(void);
void Rotate(void);
void Scale(void);
void SetUpGraphics(short *type, short *HRes,
                   short *VRes);
void TransformWorldToEye(float xw, float yw,
                         float zw, float *xe,
                         float *ye, float *ze);

main()
{ unsigned short type,mode;

  SetUpGraphics(&type,&HRes,&VRes);
  EndGraphics256();
//  GetData();
  CubeData();
  InitGraphics256(type, HRes);
  SetDefaultPalette256(1.6);
  do
  { GetCommand();
    if (strcmp(Cmd,"CG")==0) GetCentreOfGaze();
    else if (strcmp(Cmd,"EY")==0) GetEyePoint();
    else if (strcmp(Cmd,"MO")==0) Move();
    else if (strcmp(Cmd,"QU")==0) Quit();
    else if (strcmp(Cmd,"RO")==0) Rotate();
    else if (strcmp(Cmd,"SC")==0) Scale();
    else if (strcmp(Cmd,"VI")==0) DrawView();
  }
  while (TRUE);
}

void CalCoeffs(void)
 { double rxy, rxyz, costh, sinth, cosph, sinph;

char buffer[81];
   rxy=sqrt(EyeX*EyeX+EyeY*EyeY);
   rxyz=sqrt(EyeX*EyeX+EyeY*EyeY+EyeZ*EyeZ);
   if (rxy==0.0)
   { costh=1.0; sinth=0.0;
   }
   else
   { costh=EyeX/rxy; sinth=EyeY/rxy;
   }
   if (rxyz==0.0)
   { cosph=1.0; sinph=0.0;
   }
   else
   { cosph=EyeZ/rxyz; sinph=rxy/rxyz;
   }
// Coefficients of world- to eye-space
// transformation equations
   WE11=-sinth; WE12=-cosph*costh; WE13=-sinph*costh;
   WE21=costh;  WE22=-cosph*sinth; WE23=-sinph*sinth;
                WE32=sinph;        WE33=-cosph;
                                   WE43= rxyz;
// Viewing distance
   ViewDist=0.5*HRes/tan(0.5*PiOn180*AngView);
}

void CubeData(void)
{ Vertex3[0].x=-50; Vertex3[0].y=-50; Vertex3[0].z=-50;
  Vertex3[1].x= 50; Vertex3[1].y=-50; Vertex3[1].z=-50;
  Vertex3[2].x=-50; Vertex3[2].y= 50; Vertex3[2].z=-50;
  Vertex3[3].x= 50; Vertex3[3].y= 50; Vertex3[3].z=-50;
  Vertex3[4].x=-50; Vertex3[4].y=-50; Vertex3[4].z= 50;
  Vertex3[5].x= 50; Vertex3[5].y=-50; Vertex3[5].z= 50;
  Vertex3[6].x=-50; Vertex3[6].y= 50; Vertex3[6].z= 50;
  Vertex3[7].x= 50; Vertex3[7].y= 50; Vertex3[7].z= 50;
  NumVertices=8;

  Edge3[0].start=0; Edge3[0].finish=2;
  Edge3[1].start=2; Edge3[1].finish=3;
  Edge3[2].start=3; Edge3[2].finish=1;
  Edge3[3].start=1; Edge3[3].finish=0;
  Edge3[4].start=4; Edge3[4].finish=5;
  Edge3[5].start=5; Edge3[5].finish=7;
  Edge3[6].start=7; Edge3[6].finish=6;
  Edge3[7].start=6; Edge3[7].finish=4;
  Edge3[8].start=0; Edge3[8].finish=4;
  Edge3[9].start=6; Edge3[9].finish=2;
  Edge3[10].start=3; Edge3[10].finish=7;
  Edge3[11].start=5; Edge3[11].finish=1;
  NumEdges=12;
}

void DoPerspective(float x, float y, float z,
                   short *scnx, short *scny)
{ *scnx=ViewDist*x/z; *scny=ViewDist*y/z;
}

void DrawView(void)
{ unsigned short edge,s,f,screenxs,screenys,
   screenxf,screenyf;
  float xes,yes,zes,xef,yef,zef;

  FilledRectangle256(0,0,HRes-1,VRes-1,BkgdCol);
  CalCoeffs();
  for (edge=0;edge<NumEdges;edge++)
  { s=Edge3[edge].start; f=Edge3[edge].finish;
    TransformWorldToEye(Vertex3[s].x-GazeX+CentreX,
                        Vertex3[s].y-GazeY+CentreY,
                        Vertex3[s].z-GazeZ+CentreZ,
                        &xes, &yes, &zes);
    TransformWorldToEye(Vertex3[f].x-GazeX+CentreX,
                        Vertex3[f].y-GazeY+CentreY,
                        Vertex3[f].z-GazeZ+CentreZ,
                        &xef, &yef, &zef);
    DoPerspective(xes,yes,zes,&screenxs,&screenys);
    DoPerspective(xef,yef,zef,&screenxf,&screenyf);
    DrawLine256(screenxs+(HRes>>1),(VRes>>1)-screenys,
                screenxf+(HRes>>1),(VRes>>1)-screenyf,
                LineCol);
  }
  ungetch(getch());
}
void GetCentreOfGaze(void)
{ if (CmdTail!=NULL)
    sscanf(CmdTail,"%f %f %f",&GazeX,&GazeY,&GazeZ);
}

void GetCommand(void)
{ unsigned char response[81];

  FilledRectangle256(0,VRes-16,HRes-1,VRes-1,BkgdCol);
  Input256("\x0a ",0,VRes-16,TextCol,BkgdCol,response);
  strupr(response);
  Cmd[0]=response[0]; Cmd[1]=response[1]; Cmd[2]='\0';
  strcpy(CmdTail,strstr(response," "));
}
```

Figure 5 - Demonstration Program

the conventions of text handling, and insist on having the origin at the top left with the Y-axis running vertically downwards.

World to Eye

Correcting for the centre-of-gaze, the point at which the camera or eye is looking, is straightforward; all that is necessary is to subtract the centre-of-gaze coördinates from the world coördinates of each point, in effect, shifting the origin of the world coördinates to the centre-of-gaze. Moving to eye coördinates, while keeping the centre-of-gaze fixed, is more complicated. The easiest way to think about it is to work in spherical coördinates. Referring to Figure 3, let **O** be the origin of the Cartesian coördinate system (after allowing for the centre-of-gaze) and let **E** be the eye, then the spherical coördinates of **E** are given by the radius ρ and the angles θ and π. Usually, the eye will be positioned so that it looks inwards, generally in the direction of the origin. It is convenient to orient the eye coördinates with the new origin at the eye and with the eye *z*-axis running positively in the direction **EO**. The orientation of the eye *x*- and *y*-axes is then somewhat arbitrary but adjusting them to be parallel to the display X- and Y-axes is normal, giving a left-handed system. Doing the algebra (the details of which you will find well set out in Ammeraal) leads to the transformations in Equation 1 where x_{world}, y_{world} and z_{world} are understood to have been adjusted for the centre-of-gaze.

Often it is easier to think of the position of the eye in terms of Cartesian rather than in spherical coördinates. These are connected by the relationships:

$$\rho_{xy} = \sqrt{x_{eye}^2 + y_{eye}^2}$$

$$\rho = \sqrt{x_{eye}^2 + y_{eye}^2 + z_{eye}^2}$$

$$\cos(\theta) = x_{eye}/\rho_{xy}$$

$$\sin(\theta) = y_{eye}/\rho_{xy}$$

$$\cos(\phi) = z_{eye}/\rho$$

$$\sin(\phi) = \rho_{xy}/\rho$$

Finally, we need to go, in 3-point perspective, from the eye-coördinates to the coördinates on the screen, X and Y. The process is one of projection. Referring to Figure 4 we see that the triangles **EQR** and **EQ'R'** are similar. Their sides are proportional, so

$$\frac{\mathbf{Q'R'}}{\mathbf{EQ'}} = \frac{\mathbf{QR}}{\mathbf{EQ}}$$

so that $$\frac{X}{d} = \frac{x_{eye}}{z_{eye}}$$

giving $$X = d.x_{eye}/z_{eye}$$

Similarly $$Y = d.y_{eye}/z_{eye}$$

The distance *d* between the view point and the projection screen still has to be specified. There are a variety of ways of doing this. One is to leave *d* as a free parameter for the user to choose. Making *d* bigger will enlarge the image. If you move the eye position back so as to shrink the image down again the overall effect will be like that of using a telephoto lens on a camera - the perspective becomes flattened. In fact, as both *d* and the eye position tend towards infinity, all perspective is lost and an orthographic projection is reached. Reversing the process, ie making *d* small and compensating by bringing the eye in close, is like using a wide angle lens and produces similar exaggerated perspective. As a photographer, I tend to think in terms of using lenses of particular focal length to fix the perspective effect and then moving in and out to get the right size (ok, there are zoom lenses!). A

$$
\begin{aligned}
x_{eye} &= -\sin(t).x_{world} + \cos(t).y_{world}\\
y_{eye} &= -\cos(p).\cos(t).x_{world} - \cos(p).\sin(t).y_{world} + \sin(p).z_{world}\\
z_{eye} &= -\sin(p).\cos(t).x_{world} - \sin(p).\sin(t).y_{world} - \cos(p).z_{world} + r
\end{aligned}
$$

Equation 1

```
void GetData(void)
{ short i;

  printf("Number of vertices "); scanf("%i",&NumVertices);
  printf("Vertex: give x, y and z coordinates\n");
  for (i=0;i<NumVertices;i++)
  { printf("%6i: ",i);
    scanf("%f %f %f",&Vertex3[i].x,
          &Vertex3[i].y,&Vertex3[i].z);
  }
  printf("\nNumber of edges "); scanf("%i",&NumEdges);
  printf("Edge: give start + finish vertex number\n");
  for (i=0;i<NumEdges;i++)
  { printf("%4i:   ",i);
    scanf("%i %i",&Edge3[i].start,&Edge3[i].finish);
  }
}

void GetEyePoint(void)
{ if (CmdTail!=NULL)
    sscanf(CmdTail,"%f %f %f %f",
           &EyeX,&EyeY,&EyeZ,&AngView);
}


void Quit(void)
{ EndGraphics256(); exit(0);
}

void SetUpGraphics(short *type, short *HRes, short *VRes)
{
// My graphics setup
*type=2; *HRes=640; *VRes=3*(*HRes)/4;
  if (InitGraphics256(*type, *HRes)==0)
  { EndGraphics256();
    printf("Invalid mode/resolution\n"); exit(1);
  }
}


void TransformWorldToEye(float xw, float yw,
                         float zw, float *xe,
                         float *ye, float *ze)
{ *xe = WE11*xw+WE21*yw;
  *ye = WE12*xw+WE22*yw+WE32*zw;
  *ze = WE13*xw+WE23*yw+WE33*zw+WE43;
}

void Move(void)
{ if (CmdTail!=NULL)
    sscanf(CmdTail,"%f %f %f",&MoveX,&MoveY,&MoveZ);
  CentreX+=MoveX; CentreY+=MoveY; CentreZ+=MoveZ;
}

void Rotate(void)
{ short v;
  float s,c,temp;


  if (CmdTail!=NULL)
    sscanf(CmdTail,"%f %f %f",
           &RotateX,&RotateY,&RotateZ);
  if (RotateX!=0.0)
  {  c cos(PiOn180*RotateX); s=sin(PiOn180*RotateX);
    for (v=0;v<NumVertices;v++)
    { temp=Vertex3[v].y*c-Vertex3[v].z*s;
      Vertex3[v].z=Vertex3[v].y*s+Vertex3[v].z*c;
      Vertex3[v].y=temp;
    }
  }
  if (RotateY!=0.0)
  { c=cos(PiOn180*RotateY); s=sin(PiOn180*RotateY);
    for (v=0;v<NumVertices;v++)
    { temp=Vertex3[v].z*c-Vertex3[v].x*s;
      Vertex3[v].x=Vertex3[v].z*s+Vertex3[v].x*c;
      Vertex3[v].z=temp;
    }
  }
  if (RotateZ!=0.0)
  { c=cos(PiOn180*RotateZ); s=sin(PiOn180*RotateZ);
    for (v=0;v<NumVertices;v++)
    { temp=Vertex3[v].x*c-Vertex3[v].y*s;
      Vertex3[v].y=Vertex3[v].x*s+Vertex3[v].y*c;
      Vertex3[v].x=temp;
    }
  }
}

void Scale(void)
{ short v;

  if (CmdTail!=NULL)
    sscanf(CmdTail,"%f %f %f",
           &ScaleX,&ScaleY,&ScaleZ);
  for (v=0;v<NumVertices;v++)
  { Vertex3[v].x* ScaleX; Vertex3[v].y*=ScaleY;
    Vertex3[v].z* ScaleZ;
  }
}
```

Figure 5 - Demonstration Program (Continued)

.EXE Magazine, Vol 5, Issue 11, May 1991

particular lens has a particular angle-of-view and it is from an angle-of-view that I prefer to obtain d using the formula:

$$d = \frac{ScreenWidth}{2.\tan(\pi.AngleOfView/360)}$$

where the angle of view is in degrees. As a guide, the 'standard' 50mm focal length lens on a 35mm camera has an angle of view of about 45°, a 24mm wide angle about 84° and a 135mm telephoto 18°.

The perspective transformation looks deceptively simple, however it contains two traps for the unwary. If z_{eye} becomes 0.0, ie a point happens to coincide with the eye, something nasty will happen. Exactly what depends on the design of your compiler. As x_{eye} and y_{eye} should also be 0.0, the result ought to be indeterminate but certainly the case is to be avoided. Even more interesting is what happens if a point is behind the eye, so that z_{eye} is negative. Intuitively the point should be invisible, but the equations as they stand will simply invert the point. One solution to both these problems is to ignore points for z_{eye} less than some suitable small value, say *0.000001d*. When you come to draw lines and polygons, these will have to be clipped at a plane lying just in front of the eye. Alternatively, you can be cavalier about the problem and say that the eye just has to be far enough back so that all points are in front of it!

Demonstration code

The program in Figure 5 implements the viewing transformations in Microsoft C to display wire-frame objects in perspective. It makes use of the Super-VGA graphics library previously published in .EXE and uses very simple data structures to keep things as uncluttered as possible. The real business of the program is carried out by the subroutines `CalCoeffs`, `DoPerspective` and `TransformWorldToEye`. `CalCoeffs` works out the coefficients for equations 1 which are implemented in `TransformWorldToEye`. The program takes the cavalier approach to the problem of zero or negative values of z_{eye}, the excuse being that it will allow you to experiment with the problem values described above. The program recognises seven commands:

CG with three parameters to set the x-, y- and z-coördinates of the centre-of-gaze

EY with four parameters to set the x-, y- and z-coördinates of the eye location and the angle of view, in degrees

VI clears the screen and draws the wire frame object

MO with three parameters moves the object by the given amounts in the x-, y- and z-directions

SC with three parameters scales the object by the given factors in the x-, y- and z-directions

RO with three parameters rotates the object by the given angles about the x-, y- and z-axes, anti-clockwise looking inward is taken as positive

QU quits the program

where all the coördinates are world coördinates and parameters are delimited by spaces.

Using the cube data provided, suitable values to start with are:

```
CG 0 0 0
EY 300 200 100 30
```

EXE

Dr Graeme Webster was formerly Head of Department of Computer Science and Deputy Director, Academic, of Teesside Polytechnic. He has been involved with computer graphics for the last 20 years with an especial interest in 3D visualisation for Designers. Currently setting up a Centre for Scientific Visualisation under the aegis of the Teesside Development Corporation.

The code given with this article, together with the SuperVGA library, is available on disk. Send a blank floppy disk to the Editor, following exactly the instructions given on Page 1, column 1. If you do not follow the instructions, your disk will join the Editor's collection. Mark your envelopes '3D-GRAPHICS'.

Objects At Your Fingertips.

Now, if you want to develop applications for Windows 3.0, there's a fast and easier way to do it with the premier object-oriented programming language. Smalltalk/V.®

With Smalltalk/V Windows, you can explore, prototype, build finished applications and ship them runtime free.

You can tap into applications using DDE so effortlessly you don't have to be a Windows expert to do it.

And with one of the world's most comprehensive class libraries, you can choose our objects or easily build your own.

But whatever you develop, it will be portable between the Windows, OS/2 and Mac versions of Smalltalk/V.

With so much at their fingertips, more people are solving more problems with Smalltalk/V than any other object-oriented programming system.

At only £330 + VAT and no runtime charges, you can solve them, too.

Just call us at 071-436 9481. And see why programming Windows has never been easier.

Smalltalk/V Windows

COCKING & DRURY (SOFTWARE) LTD

180 Tottenham Court Road, London W1P 9LE Phone 071-436 9481 FAX 071-436 0524

CIRCLE NO. 760

The XGA Engine

The XGA is the latest in IBM's line of PC video adapters. It contains significantly more powerful hardware than its predecessors, as IBM's Nick Butler explains.

XGA is the standard video subsystem on the new PS/2 Model 90 XP486 and PS/2 Model 95 XP486 families of machines. It is also available as a Micro Channel option card (IBM PS/2 XGA Display Adapter/A) for use in 386SX, 386 and 486 Micro Channel PS/2s. The XGA has many features for graphical user interfaces, allowing for highly interactive pop-up icons, pull-down menus and other interactive components of today's applications. High-resolution screen modes (up to 1026 x 768, with 256 colours) allow more windows to be displayed on the screen at one time, and give greater text clarity. The XGA video subsystem chip set was designed and developed at the IBM United Kingdom Laboratories in Hursley, England together with the PS/2 XGA Display-Adapter/A.

The *system bus interface* and the *CRT and memory controller* manage the XGA subsystem and the screen display. They provide the system processor with direct access to the video RAM through one of three apertures: one for real mode, one for protect mode on a 16-bit processor or operating system and one for protect mode on a 32-bit processor with a 32-bit operating system. The XGA uses dual-port video memory (VRAM) to store pixel data. VRAM offers better performance than DRAM, since one port updates the displayed information while the other refreshes the screen. Depending on the country, the subsystem ships with 512 KB or 1 MB installed (it can be upgraded to 1 MB using the PS/2 Video Memory Expansion option).

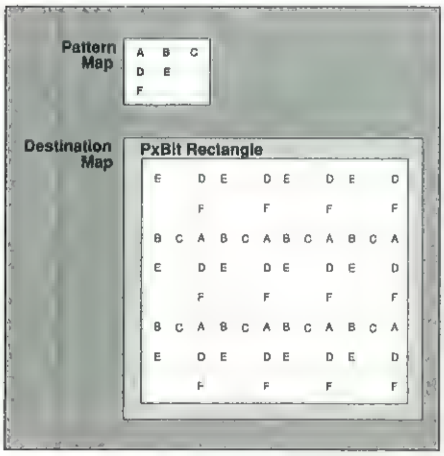


Figure 1 - Patterns are automatically Tiled

A drawing coprocessor provides a range of hardware drawing functions that operate on pixels in memory: Pixel Block Transfer (PxBlt), a line draw, an area fill, logical and arithmetic pixel mixes, scissoring, map masking, (X,Y) coordinate based addressing and a fast state save/restore. A sprite controller interfaces to the sprite buffer to display the 64x64 pixel sprite image on the screen.

Pixels, maps and addressing

The drawing coprocessor works on pixels within pixel maps. A pixel map is an area of memory at a given address (the base address) with a defined width, height and pixel format. Pixels can have 1, 2, 4 or 8-bits. The pixels can be ordered within bytes in two ways: left to right or vice versa. The XGA allows pixel maps of any arbitrary size up to 4096x4096 pixels.

The coprocessor is programmed using X,Y coordinates that are automatically converted into linear memory addresses (using the defined width and pixel size) before accessing the physical memory.

The programmer can define up to four pixel maps at one time. Three maps (A, B and C) are general-purpose; the other is always used as the mask map. When starting a drawing operation, the programmer tells the coprocessor which maps are to be used as the source, the pattern and the destination. In this way, map A, for example, could be the display pixel map. It could then be used as source, destination, or both, without having to move the pixel map parameters from one set of registers to another.

A certain amount of space is available in the non-displayed (offscreen) areas of video memory. This is often used for storing fonts and offscreen pixel maps. GUIs, however, make extensive use of offscreen pixel maps (for example, for pull-down menus), and may be called upon to use fonts that are too large to fit in offscreen video memory. When the offscreen video memory is fully used, the GUI device driver may then start using normal system memory. With previous adapters, this has generally meant processing the pixels using the system processor. This is slow compared to special drawing hardware, especially if the operation is not a simple copy. It also ties up the processor and prevents it from preparing for the next drawing operation.

The XGA uses bus-mastership on the Micro Channel to overcome these problems. The drawing coprocessor's power can be used on pixel data anywhere in system or video memory.

To support paged memory environments, where the paging unit in the 80386 or 80486 is turned on, the XGA includes its own paging unit, using page tables of the same form. The XGA can operate using the main page tables used by the operating environ-

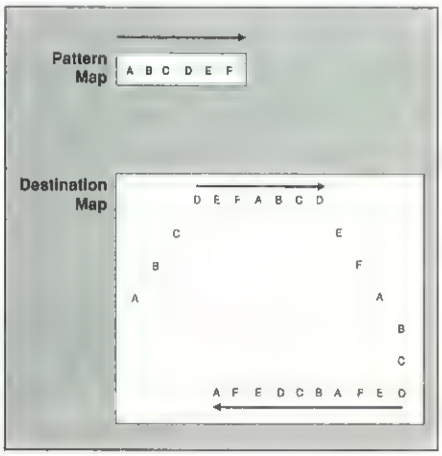


Figure 2 - Line patterns are also Pixel Maps

ment (with its cooperation), or on tables built by device drivers or applications themselves.

Pixel block transfer

The Pixel block transfer (*PxBlt*) function works with three operands: the source, the pattern and the destination. The source may come from a pixel map to copy data, or from colour registers to set the destination to a particular colour. The pattern may come from a pixel map, indirectly from the source, or it may be disabled.

For each pixel, the source and destination are combined using a Mix function selected by the 1-bit-per-pixel pattern. A 0 in the pattern selects the background mix, and a 1 selects the foreground mix. A full set of logical mixes is provided, supporting all the OS/2 and Windows Raster Operations (ROPs), with a selection of arithmetic mixes. Fast text drawing is crucial to windowing environments and other interactive applications. The pattern can select between foreground and background colours, allowing a 1-bit-per-pixel text font in the pattern map to be rapidly expanded to coloured characters in the destination.

The pattern and the source have another common feature. The X,Y addresses for these maps automatically wrap when they reach either side or top and bottom. This allows a small pattern to be 'tiled' over a large area in the destination using a single operation. The wrapping ability of the source X and Y allows multicoloured 'tiles' to be defined.

Lines, fills and scissors

The coprocessor draws lines using the Bresenham line-drawing algorithm. All pixels of a line can be drawn, or the first or last pixel can be suppressed to draw polylines correctly.

As with PxBlt, the pattern and source X,Y coordinates wrap at the edge of the maps. Line drawing, however, is different. While the X,Y coordinates of the destination move along the required path, the coordinates in the source and pattern move horizontally - left to right only. A simple 1-pixel-high pattern is drawn along the line.

Short lines can be drawn with simpler commands using 'short stroke vectors,' line segments up to 16 pixels long, and in one of eight directions (the X axis, the Y axis and the 45-degree lines in between).

The XGA coprocessor allows arbitrary areas to be filled rapidly. The area is defined by an outline drawn in a 1-bit-per-pixel map. The outline is drawn using special versions of the line drawing functions (including short strokes).

When the outline is complete, a special PxBlt fills the inside of the defined figure. A parity fill algorithm is used, where a rectangle is scanned from left to right, starting from outside the figure, and alternately moving in and out of the figure as each successive boundary is crossed. The area can be filled with a pattern if required.

Why did we use memory-mapped registers, when traditionally, display adapter registers are mapped into I/O address space

The coprocessor can, under programmer control, automatically clip or 'scissor' (ie not draw) pixels that an operation attempts to draw outside a specified area. That area can be a simple rectangle, or a more complex shape defined in a pixel map (the mask map), with a pattern of 1s and 0s (1 allows the corresponding pixel in the destination to be changed; a 0 protects that pixel). This function can be used, for example, when drawing into a background window that is partially obscured by other windows.

The hardware sprite

A hardware sprite allows a steady graphics cursor to be displayed without affecting the contents of video memory. This avoids the need for software collision detection.

If there is no hardware sprite, software must maintain a graphics cursor in the display planes, removing it whenever drawing takes place to avoid overwriting it. When the cursor is moved (for example, in response to a mouse movement), the software must remove the existing cursor image, replace the original graphics data, and redraw the cursor image at the new location. This uses valuable processor time, reduces performance and results in a cursor that flickers on and off and is unresponsive to mouse movements. The sprite in the XGA frees software from this burden and provides a steady, responsive cursor.

The hardware sprite can be any size up to 64x64 pixels. Its origin can be placed anywhere on the screen, including off the top and/or left sides. Each pixel in the sprite can be transparent, one of two programmed colours, or the 1s complement of the underlying pixel. The sprite colours are applied after the palette; each can be selected from a range of 262,144 colours.

Accessing the XGA

XGA is controlled using a combination of I/O-mapped and memory-mapped registers. I/O-mapped registers are those that appear in the I/O address space of an 80x86 processor, and are accessed using IN, OUT, or other I/O instructions. Memory-mapped registers appear in the memory address space of an 80x86 processor, and are accessed using standard memory operations with all the available combinations of registers and addressing modes. In addition, many I/O-mapped registers are indexed (that is, the register is selected using an index in one I/O port, and the data for all indexed registers is written through a second I/O port). This technique, used also by the VGA, reduces the I/O address space required.

Memory-mapped registers are generally used to control the drawing coprocessor, where frequent access requires good performance. I/O-mapped registers (indexed and direct) are used for the remainder (mainly setup registers, where performance is less important).

Multiple XGA adapters (multiple instances) can be used in a system. Each instance has an instance number and has its registers mapped at different addresses. The memory-mapped registers are located at some point within the address range C0000h and DFFFFh, the adapter ROM/RAM address space. The precise location is set by the PS/2 configuration process. When multiple XGA subsystems are installed in a system, the memory-mapped registers for all instances can be mapped within the same 8 KB block of address space. The allocation of addresses is the responsibility of the system configuration process, which ensures that there is no conflict between installed adapters (XGA or others).

The base address of the 16 I/O registers of an XGA is 21x0, where 'x' is the instance number. The base address of the

C-scape™ 3.2

The latest release of C-scape 3 — the best selling character/graphics based screen interface C library in the UK. Runtime royalty free. Includes Look and Feel Screen Designer™ and Source Code. DOS, OS/2 **£370**
Also available for Windows, UNIX and VMS.

Heron Software C-scape Extension Library

Enhanced set of functions for C-scape 3 — Price
Includes Source Code..... DOS **£99.95**
Unix **£199.95**

Phar Lap 386 | DOS Extender™

Beat the 640K DOS limit with a flat 32-bit address space on 386 processors. Voted best of 1989. Supports a wide range of 32-bit languages. .....SDK **£315**

NEW Phar Lap 286 | DOS Extender™

£315

---

db_VISTA™ III — Release 3.2

Complete family of C library dbms functions and database management utilities for DOS, Windows, OS/2, Macintosh, UNIX, QNX, VMS. Source code available — runtime royalty free.
Single-user DOS, OS/2 or Windows DBMS object libraries..... **£435**
Ask for details of other options and multi-user pricing.

Magna*Charter* II

Enhanced version of the much acclaimed Magna*Charter* charting package for Windows 3, 99x99 cells with wide range of printer support. Complete set of symbols — or add your own..... **£160**

Magna*Charter*

Still available, Magna*Charter* for DOS (includes runtime Windows 2.)..... **£118**

PERSONAL ACCOUNTANT

The only complete personal finance package including — Accounts, Assets and Liabilities and Inventory and Insurance.....DOS **£199**

All Systemstar products are distributed and supported by Systemstar. Prices are exclusive of VAT and subject to change. Orders and enquires to:

SYSTEMSTAR LIMITED
1-3 Parliament Square, Hertford, SG14 1EX
Telephone: (0992) 500919 Facsimile: (0992) 554261

CIRCLE NO. 761

SEE US AT SOFTWARE TOOLS 91
11th - 13th June
STAND 58

System Architect™

A PC based Microsoft Windows CASE tool

Microsoft Windows V 3.0 compatible

REAL TIME SOFTWARE
118-120 Warwick Street,
Leamington Spa, Warks
CV32 4QY
TEL: 0926 450858

CIRCLE NO. 810

KIBWORTH COMPUTER TRAINING

The rational choice for tuition in

C/C++ and Pascal

Our courses are an extremely rewarding experience. They are:

* Conducted in really tiny groups to achieve proven and incomparable effectiveness of one-to-one dialogue.
* Adapted wherever necessary to your particular needs. So no two courses are precisely alike, and they all teach as much as you can possibly learn in the available time.
* Devised, presented, and kept up-to-date by the same vastly experienced programmer/tutor who can assess your needs.
* Superbly equipped, and generously provided in every way.
* Offered at competitive prices.

For details of our courses, including the most efficient approach to WINDOWS 3, contact Eric Richards.

Kibworth Computer Training
68 Springfield Crescent
Kibworth Beauchamp
Leicester LE8 OLH

0533 - 792653

CIRCLE NO. 762

memory-mapped registers of an XGA is (ROM Base addr) + 7 KB + (128 x instance number).

You might wonder why we used memory-mapped registers, when traditionally, display adapters such as the VGA and the 8514/A have been controlled through registers mapped into the I/O address space. Memory-mapped registers have been introduced to control the drawing coprocessor, where accesses are frequent and good performance is essential.

The I/O address space in the 80X86 is limited to 64 KB, so individual adapters can only use a restricted number of addresses (to avoid possible conflicts between adapters). When the adapter contains many registers (typical of a display controller), indexed register addressing schemes are often used, as noted above. Memory address space is much larger (1 MB minimum in real mode), so adapters with memory-mapped registers can avoid indexing, thus allowing direct access to all registers, and reducing code space and execution time.

When the 80386 is running in protect mode, the processor normally checks I/O accesses by applications to ensure they are allowed. It reads the I/O Permission Bitmap, a process that adds 20 cycles to each individual I/O access. Memory-mapped registers avoid this overhead, reducing to one-tenth the time taken to start many graphics operations.

Another advantage relates to multiple display adapters. Each instance of the XGA has a different set of register addresses, as described above. Software must be able to run with any possible set. The 80X86 allows base-plus-offset addressing for memory access, using a segment register and an immediate or register-based offset. The software would typically set the segment register to point to the first address of the memory-mapped registers, and then have immediate pointers to specific registers. I/O addresses, on the other hand, are always contained in the DX register; no form of base-plus offset addressing is possible. For I/O-mapped registers, DX must be calculated correctly before each access, taking time and code space.

The register memory map for the XGA coprocessor (Figure 3) illustrates how all the important registers are in plain view and easily accessible.

Byte 3	Byte 2	Byte 1	Byte 0	
Page Directory Base Address				0
Current Virtual Address				4
				8
		State B length	State A length	C
	Pixel Map Index	P1 Control		10
Pixel Map n Base Pointer				14
Pixel Map n Height		Pixel Map n Width		18
			Pixel Map n Format	1C
		Bresenham Error Term		20
		Bresenham K1		24
		Bresenham K2		28
Direction Steps				2C
				30
				34
				38
				3C
				40
				44
	Dest Colour Comp. Cond.	Background Mix	Foreground Mix	48
Destination Colour Compare Value				4C
Pixel Bit Mask				50
Carry Chain Mask				54
Foreground Colour Register				58
Background Colour Register				5C
Operation Dimension 2		Operation Dimension 1		60
				64
				68
Mask Map Origin Y Offset		Mask Map Origin X Offset		6C
Source Map Y Address		Source Map X Address		70
Pattern Map Y Address		Pattern Map X Address		74
Destination Map Y Address		Destination Map X Address		78
Pixel Operation				7C

Figure 3 - Memory-mapped XGA Coprocessor registers

Programming the XGA

There are several ways to program the XGA. For environments such as Windows and OS/2, IBM provides suitable driver software. Since GUI applications are device-independent, there are no XGA-specific programming issues in these environments. If you are operating at DOS level, then IBM provides a TSR software driver with the XGA card. Its API is upwards compatible with the driver for the 8514A adapter, and is documented in the *XGA Display Adapter Interface Technical Reference* (list price £14, part no 15F2154 from your IBM dealer). If you need to go to a lower level, the XGA interface must be programmed by writing to the hardware directly; there are no INT 10h BIOS extensions. The *Video Subsystem section of the PS/2 Hardware Interface Technical Reference: Common Interfaces* (part no 84F9809) gives full details of the XGA register set. This includes useful advice for new application developers and those wishing to migrate existing applications. It should help to get the best from the XGA and to ensure compatibility with any future extensions. The *PS/2 XGA Display Adapter Technical Reference £5.00* (part no 15F2219) gives information specific to the adapter card implementation of the XGA.

EXE

Nick Butler holds a BA in Electrical Sciences from Cambridge University. He works at IBM UK Laboratories Ltd, Hursley, where he was lead engineer on the XGA chip design and development.

This article is a shortened version of a paper which appeared in IBM Personal Systems Developer, and appears by kind permission of its author and IBM.

A Light at the End of the Tunnel

Turbo Pascal is the first 3GL environment to be offered as a full Windows product. Paul G Smith has been looking at it.

To have a basis for comparison for Turbo Pascal for Windows, you need some idea of the existing, C-based approach to Windows. A baby Windows program written in Microsoft C typically consists of about six files (C source, module definition, make file, text resource file and, just for fun, a couple of resources - say an icon and a cursor). These have to be corralled into one .EXE using a motley collection of command line, character-mode interactive and Windows-based tools. Borland C++ makes great strides in smoothing this process, but the IDE still runs in character mode - so you can't bring up your program's window beside the source code that generated it - and, more importantly, you are still working with Window's rotten non-OOP, 550 function-call API. Unless you buy in a class library, or use one of the constricting 4GLs/application generators, you are still a satellite of the 90 line 'Hello World!' program.

This, then, is the environment into which Turbo Pascal for Windows (henceforth 'TPW') has been launched. Practical, full-blooded Windows programming remains the territory of an elite, who have spent time and blood sussing the system. In this age of evermore sophisticated software tools, we are overdue for something better.

Installation

TPW is dead easy to install. Assuming you have first installed Windows 3 (a prerequisite), all you do is place the first of the installation disks in a floppy disk drive and start the INSTALL program. This is a proper Windows application, running within the Windows environment. (Its display is *way* over the top, but we won't spoil the surprise for you.) The files on the installation disks are packed, to conserve space: the 5.25-inch disk version is supplied on three High Density disks, and unpacked to 3.8 MB.

Editor and Environment

The installation process creates three program icons in the Windows program manager: one for the TPW Integrated Development Environment (IDE), one for the Whitewater Resource Toolkit and one for the Debugger (which you would normally start from the IDE). To start TPW, you double-click the IDE's icon.

The Integrated Development Environment (IDE) for TPW is most impressive. The entire development process for Windows applications can now be carried out from within the Windows environment. You still get the speed and power of the TPW compiler, and the kinds of functions that were in the old text-based IDEs. The development environment is (assuming you have a reasonably powerful PC that performs satisfactorily with Windows) a whole lot nicer to use. This product is going to make users of Borland C++ (also for Windows) very jealous: they must put up with an old-style text-based IDE.

The IDE is a fully-fledged Windows application, and as such it follows the Windows Multiple Document Interface conventions. In common with other MDI-style Windows applications the fundamental component of the TPW environment is the desktop (which is itself a frame window opened over the underlying Windows desktop); up to 32 editing windows at a time can be opened, stacked, resized, and moved about within this TPW desktop. You can expand an edit window to occupy the entire desktop area; edit windows can also be shrunk down to icons on the desktop. The TPW menu bar appears below the title bar of the desktop, and a status bar is displayed at the bottom (see Figures 1 and 2). Functionality provided by the menus is very similar to Turbo Pascal 6.0.

The IDE editor implements 'undo' and 'redo', which means that you can step back

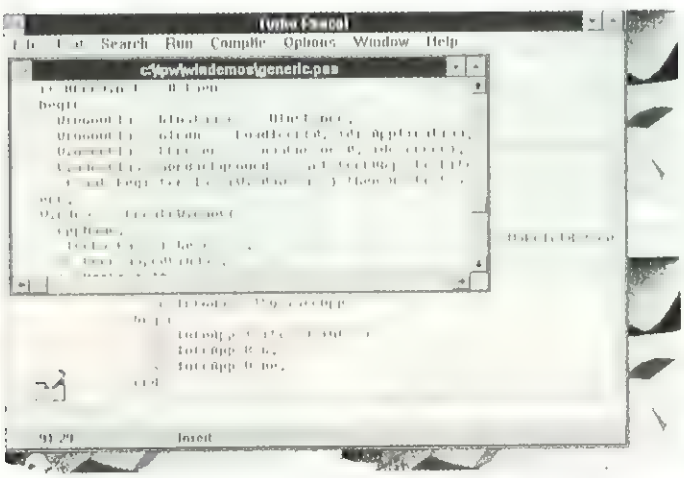


Figure 1 - The Turbo Pascal for Windows IDE

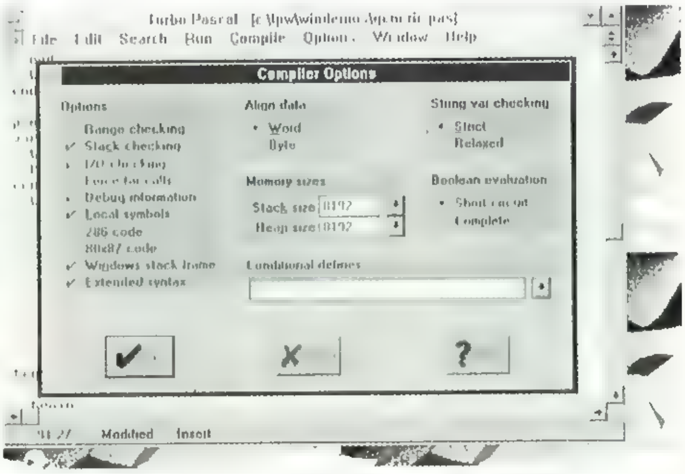


Figure 2 - Compiler options dialog box

and forth through your editing operations to recover from any accidental changes or deletions during the current edit session.

TPW's editor operates in two modes: a mode that is compatible with other Windows applications, implementing the 'Common User Access' (CUA) command set used by other Windows text editors; and an 'Alternate' mode that implements a command set more like (but not the same as) other Borland language products. You choose between the modes using the 'preferences' dialog box, which is reached through the 'options' menu. The alternate mode is interesting, because it is defined by a special macro command file that you can edit and recompile yourself to create new key bindings and editor commands.

I'd like the editor a lot more if it understood the Pascal syntax enough to, if the user wished, auto-indent source code and highlight Pascal keywords. Not a big deal, I admit, but Macintosh (Think) Pascal programmers have had that kind of facility since 1984. Syntax-directed editing isn't all that difficult: what do you say, Borland?

The debugger can be invoked from the TPW IDE: if, instead of asking it to run your program you select the 'run' menu 'debugger' command, the program will (if necessary) be recompiled and the debugger will then be invoked. Beware: the debugger uses the most recently saved version of your program, and if you have made changes to it without saving the debugger won't find out. This confused me a bit the first time I tried out the debugger: a bug in my program didn't appear in the version I tried to debug. You can configure TPW to auto-save files before running or debugging them; I'd have liked to see a warning appear when it wasn't configured that way.

TPW includes the 'Whitewater Resource Toolkit', an application that lets you construct and edit resources for your Windows application. The keystone of the Resource Toolkit is the Resource Manager, which you use to manage resource files, and to start the various resource editors. A resource editor is supplied for each resource type. These editors are graphical, allowing you to type, draw, and colour-in your resources previewing the results on screen. The Whitewater Resource Toolkit, although not integrated in any way into the IDE, is very friendly; it even uses the Windows 3 help system.

Syntax Extensions

The TPW compiler is packaged specifically as a tool for developing Windows applications. Although it may be that you could force it to build non-Windows text based applica-

BLINKER™ TAKES CLIPPER™ OUT OF THIS WORLD

Blinker eliminates the need for complex overlay structures for both Clipper Summer '87 and 5.0

* Significantly reduces run-time memory requirements
* Accelerates your development cycle 3-10 times faster than any other current linker and even faster when linking incrementaly
* Dynamically overlays C and Assembler code sich as Extend.lib, dGE, FUNCky, other Clipper Add-ons and your own 'C' code
* Optimises your application with advanced profiling functions
* Includes memory defragmentation, 'burning-in' of Clipper environmental variables/serial numbers and creation of demonstration versions

IT'S ALL PLAIN SAILING WITH BLINKER

What the experts say

Here are just some comments from the 5000 happy users to date

"It's blinking fast!"

"Congratulations – we gained over 100k"

"It's so good, I went through all my programs and re-compiled/linked just for the pleasure of seeing all the free memory"

"It's fast! Handles overlays like a dream"

"Amazing" "No more messing about with overlay structures!"

"Brilliant!... But my coffee breaks have been shortened!"

"Actually exceeded our most optimistic performance hopes! (Just for once)"

European sales and support:

QBS Software Limited
10 Barley Mow Passage, Chiswick, London W4 4PH
Telephone 081-994 4842. Fax 081-994 3441. BBS 081-747 1979

Authors and publishers:

Blink Inc Limited
Raglan House, Llantarnam Park,
Cwmbran NP44 3AB

CIRCLE NO. 765

tions (I found no documentation on how to do that) it looks like Borland wants you to buy a separate copy of Turbo Pascal 6.0 for text-based applications development.

As well as using the TPW compiler from within the IDE, you can invoke it from the DOS command line. All the facilities of the IDE compiler are available; you can control them using command line switches.

Windows data objects can be allocated in one of two heaps: the global heap, shared between all Windows applications, and the local heap which lives inside your application's data segment. All TPW heap allocations are made within the global heap; since there is a system-wide limit on the number of objects within the global heap, TPW sub-allocates smaller objects within larger memory blocks, which it pre-allocates and manages. Because of the way that TPW does this (it creates 'fixed' memory blocks) your application may suffer if run in the Windows 'real' mode: the way to get around this is to directly call Windows' memory manager if you are writing a memory-intensive application that is to use real mode.

Windows uses C-style null-terminated strings, unlike those familiar to most Pascal programmers. TPW extends the Pascal syntax to support arithmetic operations on pointers to arrays of characters; you can also use array-access operators on character pointers. A unit of null-terminated string handling functions is provided, so that you can confine all your string manipulations to null-terminated strings. Beware that Pascal-style string handling functions cannot be performed on null-terminated strings, so if the Pascal-style operations are dear to your heart you'll have to convert back and forth between the two formats.

Dynamic methods

In previous articles we have commented on the problems associated with 'flat' virtual method tables (VMTs used by Borland to dispatch virtual methods): if virtual methods are overridden in descendant object types, all the method table entries have to be copied into the new object's method table. If objects contain many methods,

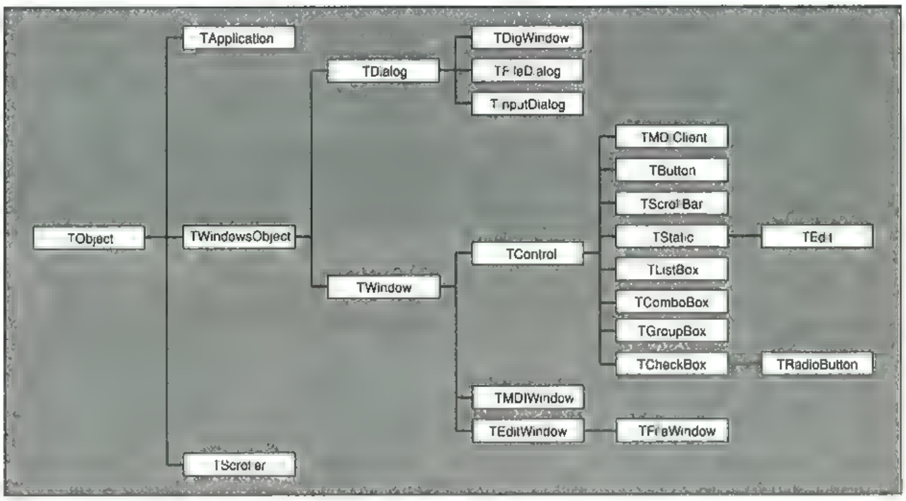


Figure 4 - ObjectWindows hierarchy

and if many objects need to be overridden (perhaps changing a small subset of the methods each time) the application data segment that holds all the method tables can quickly overflow.

Windows operates by dispatching messages; the ObjectWindows class library is designed to support the Windows message passing system. A Windows application may have to deal with lots of different types of message; the ObjectWindows class library, together with the extra definitions and implementations in your program, contains a great many different virtual methods. ObjectWindows thus defines several class types, each with many methods. Applications developed using ObjectWindows are implemented by selectively override these methods, defining new derived classes that encapsulate application-specific functionality.

If TPW only supported the 'flat' virtual method tables, in common with its predecessors Turbo Pascal 5.5 and 6.0, many ObjectWindows applications would run out of space. Borland has got around this (and at the same time provided a pretty neat way of dispatching method calls that handle Windows messages) by defining a new type of virtual method: the 'dynamic' virtual method.

Windows messages are identified by index numbers, which are integer constants. The ObjectWindows class library is organised on the basis that each message has a method, in an object type, to handle it. The `virtual` keyword is expanded, in class declarations, to include the index number for the corresponding message, which must be unique for all methods in the class and its derivations - see Figure 3.

A new type of method table has been created for dynamic methods: the dynamic method table or DMT. Each class may have either one or both of a flat VMT and a DMT, depending on the mix of virtual and dynamic methods defined for it. DMTs, unlike VMTs, contain a list of method index numbers (the Windows message indexes) followed by a list of corresponding method addresses. Only those methods that are overridden in a particular class need to have a DMT entry, which means that method table space is limited, at the cost of an iterative look-up of method entries in DMTs.

It's interesting to see Borland accepting that the space problems caused by storing flat VMTs in the application's data segment make that technique unsuitable for Windows (see previous .EXE articles). DMTs are indeed slower than VMTs, but since you can't fit all Windows application's message handlers into VMTs there's no choice but to go for a different implementation.

Dynamic methods are nice for implementing message handlers in Windows applications, in which each message has a defined index number in common with things like menu definition resources. They are not quite so nice if you have other methods that could also benefit from the reduction in VMT space, because you have to dream up unique numbers for all new dynamic methods. This is no great problem for Windows message handlers, but a real kludge in other cases, since the index number has no other relevance to the program being developed. It would be much better if programmers

```
type
  TCommandHandler = object (TEventHandle)
    commandID:     INTEGER;
    commandState:  CmdState;
    constructor Init;
    destructor Done; virtual;
    procedure doit; virtual cm_First + cm_Redo;
    procedure unDoit; virtual cm_First + cm_Undo;
  end;
```

Figure 3 - Dynamic virtual methods

could use a compiler directive to say that certain virtual methods should use DMTs, and that the compiler should auto-allocate the DMT index numbers in cases where they are left unspecified. It would also be nice if the programmer could (optionally) specify that VMT and DMTs should be allocated in far memory and take up more space than allowed at present.

Libraries

ObjectWindows is the class library bundled with TPW, and it's a beaut. It hides dozens of tedious function calls and data structures which clog up Windows programs written in C, leaving the programmer free to concentrate on the task in hand. Figure 4 shows the main Object Windows hierarchy. All classes are derived from `TObject` (necessary for the use of Borland's `TCollection` and `TStream` classes, which supply a general-purpose storage system and object persistence). The `TApplication` class handles general initialisation, such as registering the main Windows procedure, and the message dispatch loop. All this is hidden; the programmer has merely to subclass it for his own application and instantiate it. All the other classes on the tree represent visual objects: windows, dialogs and controls.

You create your application's main window by inheriting from `TWindow`. TPW's facility to associate messages with particular methods means that you do not have to construct those giant `case` statements which are a hallmark of Windows applications. If your application needs to act on `wm_Paint` messages, you simply override `TWindow`'s `Paint` method (TP6 aficionados: equivalent to `Draw`). Because a lot of the work can be done by calling object methods rather than API functions, there are fewer multi-parameter calls. You don't need to pass a window handle to a method of a window object - it's in the instance data.

The function of the various classes is reasonably obvious from their names. Specialist classes include a file selection dialog box `TFileDialog`, memory buffer and file editors (`TEditWindow and TFileWindow`), and support for the MDI system of child windows (`TMDIWindow`, `TMDIClient`) - of limited use, perhaps, since MDI has been dropped from the latest version of the CUA specification.

Users of Turbo Pascal V6.0 and the excellent Turbo Vision system will want to know: How close is ObjectWindows to Turbo Vision? The answer: quite close, but not touching. Some differences are inevitable. Turbo Vision runs in character mode and has complete control of the machine; for example, it handles its own events. ObjectWindows runs in graphics mode, and must fit in with Windows' way of doing things. So, although a lot of the identifier names are the same, TP6 applications will require some work to get them going under TPW.

Incidentally, there seems to be controversy over who wrote ObjectWindows. A Whitewater Group spokesman told .EXE that ObjectWindows was essentially the same library that Whitewater has been shipping with its Actor system for five years, adapted by Borland to be compatible with TPW - he estimated that 90% of the code was Whitewater's. The next release of Actor, he said, would contain a library on 'the same level' as TPW's. Borland, when asked about this, stated that it was really a Borland product, and that Whitewater had only supplied consultancy on the project. Hmm.

Other libraries: the new `Strings` unit, touched upon above, contains 26 functions and procedures for manipulating and converting C-style null terminated strings. A syntax extension permits the use of type `PChar` to be used in the same way as C's `char *`. For example, given the declaration

```
var P : PChar;
```

you can write

```
P :='Null terminated string';
```

This enables you to abandon Pascal-style strings altogether, which might be just as well.

The `WinCRT` unit supplies a text file device driver that redirects standard I/O to a scrollable window containing a 'virtual' character screen. Old-fashioned `ReadLn/WriteLn/GotoXY` style programs can be swiftly ported to Windows, simply by including a `uses WinCRT` at the top of the code - without need to use ObjectWindows. Of course, clever stuff, like colour manipulation, doesn't work, but this feature is extremely handy for knocking out quick-and-dirty test programs.

DLL externals

Windows allows applications to share separately built modules of codes, called'dynamically linked libraries' (DLLs). Each DLL can contain, sévèral different functions. As long as the calling interface remains unchanged, DLLs can be altered and rebuilt without requiring any changes to the client applications. TPW allows you to call functions in DLLs and build DLLs of your own. Each function in a DLL that is callable from outside the DLL is given an index number and a name. The calling application can refer to the DLL function by name (which defaults to the function's identifier) or by index number.

Figure 5 shows an example of a DLL (lifted from Borland's TPW Programmer's Guide). The entries in the exports clause, at the end, can be annotated with a `name` directive (to specify the entry's name) and a `resident` directive (to specify that the name information should be cached in memory to save look-up time). If an application calls a function in a DLL, it declares it using the `external` keyword, eg:

```
function Min(X, Y: Integer):
                          Integer;
   external'MinMax' index 1;
```

The name of the external function, if it differs from the function identifier, can be specified by adding a `name` directive. The `index` and `name` directives are both optional.

The Debugger

Turbo Debugger for Windows is a bit disappointing. It still runs in text mode: every time you hit a break point, the image on the CRT tube bounces around for a few seconds, trying to cope with the mode switch. Windows specific features include special heap inspection tools and trapping of Windows messages. For more details, please see Dan O'Brien's article on Borland C++ in .EXE March'91. There is nothing wrong with Turbo Debugger - it's a good tool - but it is eclipsed by the rest of the package. Doubtless it will be revamped when Win-

```
library MinMax;

function Min(X, Y: Integer):
              Integer; export;
begin
  if X < Y then
    Min:= X
  else
    Min:= Y;
end;

function Max(X, Y: Integer):
              Integer; export;
begin
  if X > Y then
    Max:= X
  else
    Max:= Y;
end;

exports
  Min index 1;
  Max index 2;

begin
end.
```

Figure 5 - A simple DLL

dows 3.1 appears later this year - the new version of Windows will contain special debugging hooks.

Documentation

TPW comes with six manuals: an installation and tutorial *User Guide*, a *Programmer's Guide* that defines the language and runtime library; a *Windows Programming Guide*, a *Windows Reference Guide*, a manual for the *Whitewater Resource Toolkit*; and a *Debugger User's Guide*.

The manuals are detailed and comprehensive, but they appear to have suffered a little from hurried preparation: a few typographical errors remain. You have to take a some of the program code examples with a pinch of salt. I'd like to have seen some cross-references between manuals. I found at times that I was hunting from manual to manual for something: a consolidated index would be really useful.

A few things, like editor macros, are not described at all. For information about these you'll have to look at the disk-based documentation and READ.ME files. This is a continuation of a Borland trend with its language utilities. It won't worry many users, although it's surprising that there's no hint at all of the editor's extensibility in the printed documentation.

From all parts of TPW you can call up context-sensitive on-line help, which uses the Windows hypertext help system. This is a real pleasure to use. The on-line help isn't complete, although it does contain everything important. For the more obscure stuff you still need to look elsewhere.

Conclusions

The difference between writing Windows applications in straight C and using TPW is comparable to the difference between writing straight MS-DOS programs in assembler and a high-level language. As with our review of Turbo Pascal 6.0, it has been hard to avoid gushing.

On the downside, the Pascal language is showing its age: in TPW we see more new features shoehorned into small syntactic spaces. Some of the language constructs in TPW look a teensy bit contrived. Borland is presently submitting its proposals for changes to the ANSI standard Pascal language: I hope that the ANSI committee is able to represent the wider world's requirements for language constructs that are not specific to the problems posed by specific run-time environments.

Turbo Pascal for Windows is very good indeed, continuing the trend established with Turbo Pascal 6.0. I don't know whether Borland's evident productivity improvements are due solely to their apparent wholesale adoption of OOP techniques, but it would be nice to think that it is. If they continue to crank out new and improved versions at this rate, we can look out for some pretty special stuff in future versions of Turbo Pascal.

EXE

Paul G Smith has been dragged away from product development to write this review (he needs the money). He has recently launched a range of communications products for the Macintosh, and he also consults on graphics, communications, and object oriented programming techniques. He can be contacted via CIX and AppleLink as "pgsmith".
Turbo Pascal for Windows costs £149.95. Registered owners of any Borland language may purchase it for £69.95.
Borland UK is on 0734 320022.

Willow without Weeping

WLO - the Windows Libraries for OS/2 - is a quick method of getting your Windows application running under OS/2 PM. And even if you don't give a fig for OS/2, it's good preparation for Windows' future. Andrew Marshall explains.

Originally this article was going to describe how you could increase the market for your Windows product by using Windows Libraries for OS/2 (WLO, pronounced 'willow') to get your application running under OS/2, but after the Windows Developers Conference in February, things have changed a bit. Well, quite a lot actually. Getting your application running with the WLO libraries should now be a part of your normal development cycle regardless of whether you intend to ship an OS/2 product or not. A program that is WLO-conformant will port a lot more easily to WIN32, the 32-bit Windows API, when that arrives. Given this increased profile of WLO, it is important to understand what it is, how it works, and how to integrate it into the development environment.

So what is WLO?

The WLO Development Kit, previously known as Porthole, BCL or the SMK, is a set of software tools which will allow you to build OS/2 PM applications from Microsoft Windows 3.0 compatible source code. The PM application produced by this process will run with OS/2 Version 1.21 or later. The WLO 0.9 Kit produces a separate executable file from the Windows executable. WLO 1.0, available in June, will produce a single executable which will run on either Windows or OS/2.

WLO works because the executable file formats for Windows and OS/2 are largely the same. The resource format of Windows is a subset of the OS/2 format. All that's needed is to fix up the Windows API external references. At the heart of WLO is a suite of DLLs which reproduce the functionality of the Windows DLLs in what Microsoft call 'the mapping layer' (see Figure 1). These DLLs map the Windows API calls in your program onto their PM equivalents. Certain API calls, such as `MessageBox`, have a direct PM equivalent - in this case, `WinMessageBox`. Other API calls, such as the cursor information in `RegisterClass`, require the mapping layer to preserve information across many OS/2 calls. WLO will also generate multiple Windows messages for one OS/2 message. For example, character handling requires just one message in OS/2 - this is translated by WLO into the multiple message expected in a Windows application.

The mapping layer can support multiple converted Windows applications. WLO applications can communicate with native PM applications via the Clipboard and DDE. The mapping layer automatically converts between bitmap formats for the clipboard, and window handles, memory and so on for DDE. You cannot talk to a PM window directly because of the fundamental difference in the size of the Window handles: 16-bits for Windows, 32-bits for PM.

A WLO application can load and display Windows compiled resources without a problem. Tools are supplied to convert fonts and icons to their PM equivalent formats. WINHELP.EXE is also supplied with the WLO kit so that help created for the Windows application will run unchanged (OS/2 Help is entirely different). There are some restrictions on what a WLO application can do. Most, if not all, of these restrictions are due to facilities not available under the OS/2 operating system. The main restrictions are:

- There is no sound, palette, WINMEM32 or debug API support.
- Interrupts cannot be called directly.
- There is no support for 3rd party Windows device drivers.

A Windows API function that is not supported is 'stubbed out' to use Microsoft's words, meaning that a call to a sound function, such as `OpenSound` does nothing - but it will not hang the system. Direct calls to Int 10h will cause a Protection Violation or 'Trap D'. This is because PM applications run at ring 3 of 80286 protection scheme, where as Windows applications run at ring 0 or ring 1. `Dos3Call` and `NetBIOSCall` should be used instead .

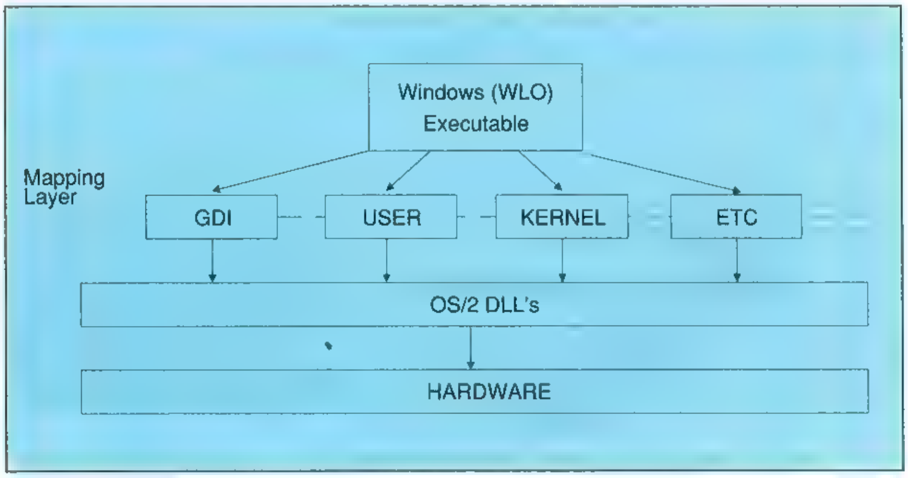


Figure 1 - The WLO Mapping Layer

5 out of 5 hackers prefer other software protection methods to Hardlock E-Y-E®

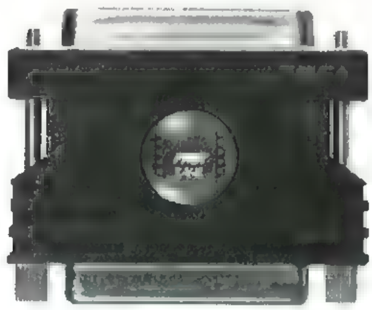
What hackers dislike...
Hardlock E-Y-E was designed using cryptographic principles. It took the experience and know-how of Germany's No. 1 in software protection and the leading edge technology of a US semiconductor company to create the ultimate software protection tool. Hardlock E-Y-E is based on a custom chip featuring secure algorithmic response rather than simple bit swapping or counting schemes.

What software developers like...
Hardlock E-Y-E combines all the features software developers require in a single product: algorithmic response to provide security and an optional non-volatile memory to allow custom configurations. FAST Electronic has made implementation of Hardlock E-Y-E in your software easy. Use HL-Crypt to protect .EXE or .COM files, or incorporate high level language interface routines in your software. The algorithm parameters and the contents of the memory can be programmed in seconds using our Crypto-Programmer card. This unique card guarantees that no one else can burn your original codes. Simply plug the card into any PC slot and start up your own Hardlock E-Y-E workshop.

What your customers will like...
Hardlock E-Y-E allows unlimited backup copies. The device is shipped with the software for the user simply to plug into the parallel interface and forget.

Daisy chainability, outstanding reliability (no battery is needed), and the most compact High-Tech design ensure that your customer will accept Hardlock E-Y-E.

What your accountant will like...
Hardlock E-Y-E needs no factory coding. This ensures optimum delivery schedules and stock flexibility. Revenues will go up as software piracy and multiple usage are prevented. Despite its wealth of features, Hardlock E-Y-E's prices remain competitive.



Hardlock E-Y-E
programmable, algorithmic response and memory option – all in one.

...As more and more software developers, customers and accountants appreciate the Hardlock E-Y-E device, hackers like it less and less.

MAGNIFEYE

Order your demo unit today. Contact Magnifeye,
235-239 Walmer Road, Walmer Studio #6, W11 4EY, Telephone 071 221 8024, Fax 071 792 3449.

CIRCLE NO. 769

What you need

WLO programs can be built in either the DOS or OS/2 environments. My advice to you would be to use the OS/2 environment, since it is only possible to debug the converted application with OS/2's Protected CodeView. You can run Windows in Real mode in the DOS compatibility box if you need to check that any changes you make still work in this environment.

On top of your normal Windows development environment, (386-based machine, at least 2 MB RAM, 30 MB hard disk, Microsoft compatible C compiler, Windows SDK, second monitor for debugging) you will need the following:

- OS/2 Version 1.21 or above. I would recommend OS/2 1.3 since it only requires 2 MB RAM.
- An OS/2 Developers Kit (IBM's or Microsoft's).
- Another 10 MB of disk space.

WLO itself can take up to another 7.5 MB depending on the installation options you have chosen. A full installation includes sample source, Windows System fonts, international copies of the WinHelp database, DLLs, import and static libraries for all models, WLO utilities, and Quickhelp format on-line documentation. WLO applications can be produced without OS/2, but this is not really recommended as it's a bit like producing Windows applications without having a copy of the Windows run-time!

You should install WLO from OS/2 since the OS/2 Installation command file will then copy across the OS/2 DLLs and a directory of what Microsoft calls its Applets. These are WLO versions of all your favourite Windows accessories - including Solitaire. Microsoft says this is to allow you to demonstrate the utility of the WLO Kit. All models, with both emulated and alternate maths libraries, are available - small and medium normally suffice. Once installed, you need to set your LIB and PATH environment variables to point to the appropriate directories. If you are going to compile a WLO application under OS/2 you will also need to add all your standard Windows environment settings. The WLO DLLs are automatically copied to the same directory as PMWIN.DLL, so they will be ready to run straight away. If you rely on any of the standard windows system fonts, these should be installed now via the OS/2 Control Panel.

It is possible using the WLO kit to have something up and running on OS/2 in around 2 days

Converting your application

Let me state straight away that if your program uses any internally developed or third party device drivers, TSRs, hardware dongles (with their assembly language checking routines), your application will not be completely converted by the WLO kit without these sections being completely rewritten for OS/2. If your application does not fall into this category you will find that you should have something running in two days or less. The shortest time I have seen for a WLO conversion is three hours. So, once you have installed the WLO kit, and after you have stopped playing Solitaire, you can attempt to convert your first application. I would recommend compiling the Sample programs supplied (GENERIC and REVERSI) before attempting your own application. A successful compile of these programs will demonstrate that the environment variables have been set up correctly. To convert your application, all you have to do is change the link statement in your makefile to use the WLO import libraries instead of the Windows ones. Figures 2a and 2b show the changes that are necessary.

```
link appname /NOD ,, libw slibcew, appname.def
```

Figure 2a - A normal Link Line

```
link appname /NOD ,, libmk_b slibcemk os2, appname.def
```

Figure 2b - A WLO link line

Now delete the executable and re-make the application. The resulting executable can be run from the OS/2 command line just like any other OS/2 program. The first thing you will notice about your WLO application, if it runs at all, is that it takes a long time to display its window. This is because of all the DLLs the OS/2 needs to load to get a WLO application running. Once running Microsoft states that a WLO application is normally around 0 - 10% slower than the same application running in native Windows. In my experience, graphics tends to be the slowest area.

If your WLO application 'traps' - that is, causes a Protection Violation - it could be for three main reasons:

- You are performing some form of direct I/O.
- You have not yet converted your DLLs.
- You've got a bug.

Initially you should comment out all direct I/O or interrupt calls, just to allow the main body of your program to run. You can convert these to `Dos3Call`or the OS/2 API equivalents later.

All DLLs that the application calls have to be converted as well. These require a little more work than the executable. OS/2 and Windows expect different register settings on initialisation of the DLL. The OS/2 DLL loader will set the registers to the OS/2 DLL initialisation values, so before the Windows initialisation routines can be called some modifications are needed. Fortunately, the WLO Kit provides a utility and a piece of code that does this job for you. The utility is called CONVDLL. It adds some code to the start of your DLL to manipulate the registers. It also adds some code to handle the unlinking of the library once all the WLO apps using it have closed. To allow CONVDLL to do its work, you must reserve space in LIBENTRY.ASM by including the file CONVDLL.INC. If you've rejected LIBENTRY.ASM and use your own assembly startup routine, now is a good time to go back to it. Figures 3 and 4 demonstrate the code and make file changes that are necessary.

Once the DLL has been compiled and linked, you can run CONVDLL. An important point to note here is that OS/2 cannot guarantee the order in which DLLs will be loaded, so your DLL may be initialised before the Windows system DLLs (USER, KERNEL, GDI etc). This means that you should not call Windows functions in your initialisation routine until after the `INCLUDE CONVDLL.INC` statement.

As for bugs: converting an application to OS/2 using WLO is a very good way of finding subtle (and not so subtle) bugs in your application. The protect mode environment of OS/2 is a lot less forgiving than Windows. Running CVP (try to get V2.3 or above: the V2.2 that came with C V5.1 has problems with WLO) will normally find the problem. I teach a Windows to OS/2 PM migration course, and I generally find that most of the conversion problems are due to bad coding practices in the Windows program. The conversion actually helps to clean up the Windows program that is being converted!

Multi-tasking issues

OS/2 is a fully pre-emptive multi-tasking operating system. Windows is not. The multi-tasking under Windows is really a round-robin scheduler. A Windows application will only lose control when the message queue is empty on a call to GetMessage or PeekMessage. This allow the Windows programmer to make assumptions about the way that shared resources can be accessed which would not be valid under OS/2. WLO protects converted applications from this and guarantees that a converted application will only be suspended to run another converted application when it calls GetMessage or PeekMessage. The rest of the PM applications are scheduled as normal. This has important implications if you decide to extend your application by the use of an OS/2 DLL. You should use semaphores to synchronise access to the shared resources inside the DLL because the DLL, being an OS/2 program, will be pre-emptively scheduled.

WLO utilities

There is a number of utilities that come with the WLO kit. CONVDLL I've already mentioned. The others are :

CONVICON, which converts Windows icons to PM format. When you install a program in the Program Manager in OS/2, the system looks for a PM icon in either the program's extended attributes or in the program's resources. Since your WLO application is really a Windows program, none will be found. CONVICON allows you to convert your Windows icon to a PM icon. To use, type:

```
CONVICON <winicon.ico>
                    <pmicon.ico>
```

WLOINST, which installs WLO DLLs or attaches PM icons to an executable as an extended attribute. The source to this utility is provided as part of the WLO kit. Microsoft recommends that you ship this utility with your product to copy over automatically the WLO DLL set onto the target system.

```
cProc LibEntry, <PUBLIC,FAR>
cBegin
```

Figure 3a - A portion of LIBENTRY.ASM

```
cProc LibEntry, <PUBLIC,FAR>
INCLUDE convdll.inc
cBegin
```

Figure 3b - LIBENTRY.ASM modified for WLO

The other function of WLOINST is to add an icon (perhaps one converted with CONVICON) to the extended attributes of a file.

WLO applications can communicate with native PM applications via the clipboard and DDE

CONVFONT is for converting fonts to PM format. Windows and OS/2 use different font file formats. This utility converts between the two.

WINHELP is the standard Windows help engine. WINHELP functions in exactly the same way as it does in Windows - it even says it is a Microsoft Windows application in its About Box. The WLO kit provides international versions of the help engine in 10 non-english, European languages. WINHELP, by the way, is a WLO application.

RCPM is the PM Resource Compiler, renamed to prevent conflicts with the Windows RC.EXE. This used to be the only clash between the Windows and OS/2 SDKs - both had a resource compiler called RC.EXE. You couldn't simply rename one since there were hard coded references to other programs. The solution I used to use was to patch one of the compilers using debug and then rename it. Microsoft has now provided a cleaner solution by supplying a new version of the PM resource compiler called RCPM. Now all you have to remember is to alter all the PM make files to use it (users of automatic code generators, such as CASE:PM, beware!).

WLO09.HLP is a quick help database containing on-line documentation for the WLO kit. This is not really a utility, but I consider Quickhelp to be an essential development aid.

Odds and ends

Two more notes. Applications will have to modify their set-up because the equivalents of WIN.INI in OS/2, OS2.INI and OS2SYS.INI, are binary files. The Windows profile functions are translated to their PM equivalents and WLO is intelligent enough to recognise some of the standard sections for example, colours, and convert the call to its PM equivalent. All other sections will 'pass through' and not be recognised. Because OS2.INI is a binary file your set-up should make any modifications that are necessary itself; the user cannot make any modifications by hand. The solution is to have a private profile for your application : this will still work using the WLO kit.

There are differences in the way that certain graphics call operate under Windows and OS/2 so WLO applications can be slightly different at the pixel level from the original Windows application.

Extending an application

It is possible to extend a WLO application to make use of OS/2 specific functions - the most important being multi-threading.

```
link dllname libentry /NOD, dllname.dll,, sdllcew libw, dllname.def
```

Figure 4a - A DLL Link line

```
link dllname libentry /NOD, dllname.dll,, sdllcemk libmk_b os2, dllname.def
```

Figure 4b - DLL link line for WLO

There are three ways to extend a WLO application:

- Adding the new code inline.
- Adding the code in a separate object module.
- Adding the code in a separate OS/2 DLL.

I would recommend the use of the last two options, since adding code inline can cause problems with prototypes and type definitions. It is not possible to include both OS2.H and WINDOWS.H in the same source module because they define the same types differently. For example, HWND is defined as being 16-bit in Windows and as 32-bit in OS/2. If you only need to call one OS/2 API routine in your code you should prototype it at the start of the module and then use it. A more preferable route would be to have a separate source file for the OS/2 specific parts of your code. This will allow you to include OS2.H in this module and have all the correct prototypes and type definitions available. All references to OS/2 functions will be resolved at link-time by the OS2.lib import library. Adding an OS/2 DLL to your code is required if the OS/2 additions need to use resources of their own. Access to this DLL has to be synchronised between multiple running instances of your converted application. Microsoft suggested that this could be done using semaphores. One of the WLO samples, OS2DLL, demonstrates how an OS/2 DLL can be called from a Windows program.

If you want to use the multi-threading capabilities of OS/2 from a converted application, there are two severe restrictions:

- Secondary threads cannot call any Windows API function except PostMessage.
- A secondary thread can only call C run-time functions if the thread code is in a DLL. It is possible to call OS/2 API functions such as DosOpen or DosClose.

These restrictions make adding multi-threading to your code all but impossible without a major rewrite.

Conclusion

The Windows Libraries for OS/2 are a very useful tool to help you move your application from Windows to the OS/2 environment. It is, however, just a tool. It won't be possible to just recompile you application and have it running under OS/2. Most serious applications will run into one or more of the restrictions outlined above. To overcome this, some sort of re-write will be necessary. But then again, many of these re-writes will save future problems with Windows V3.1 and beyond.

Version 0.9 of WLO is available now, from Microsoft UK for £100 approx or $150 if you have a contact in the U.S. Version 1.0 will be released as an upgrade - some time in June, says Microsoft. Version 1.1 will form part of the Windows 3.1 SDK when this is released later this year.

EXE

Andrew Marshall is the director of a computer consultancy company specialising in the areas of Windows and OS/2 Presentation Manager. He also is a part-time consultant for QA Training where he teaches both the OS/2 Presentation Manager and the Windows to OS/2 PM migration course. Andrew can be contacted on 0462 451496 and on Compuserve as 100016,3504. Microsoft UK can be contacted on 0734 391123.

Why programming is hard

Even with a well-defined problem to be solved, it is very difficult to produce a program that works without any bugs at all. Jules May explores the reasons why this is so.

I believe that the most spectacular example of language design is the specification for Pascal. It is good, not because it is a general language, nor because it forces (alleged) good design on the programmer, but because it solved a particular problem admirably.

Pascal is a product of its time. During the 1960's, what is known as the 'software crisis' developed. Engineers had become familiar with the capabilities of computers and started designing ever more ambitious software projects. Unfortunately, these projects were dramatically in advance of what the programmers could do, and more and more software was shipped which was so bug-ridden as to be unusable.

Out of this mess developed most of the disciplines we know today - software engineering, structured programming (which is usually taken to mean top-down structure) and the study of human factors. Software has become far more reliable, and this has allowed programmers to develop their skills still further. The only problem is, history is poised to repeat itself. The introduction of graphical environments (which are, to all intents, fully-fledged operating systems) has thrown a major spanner into the works, because not only is it extraordinarily difficult to get anything to work with them, but the programs that are being produced are bound to influence the design of programs on text-based systems. A Windows program written in C looks more like assembler on the old mainframes than a modern language on a shiny new operating system; there is enormous quantities of initialisation code and housekeeping and relatively little real work. What is going on in this example is this; top-down programming is not an appropriate programming model for Windows. In this case, textbook programming will not make a working program.

Limits of top-down

Rather than concentrate on this one example, I want to use it to look at a more general problem. The fact is that structured programming is an appropriate strategy in very few problems (although it is extremely appropriate in sections of nearly all), and it is unreasonable to try to shoehorn every possible programming task into a structured model. I suspect that one of the reasons for C's success is that, while it will almost allow true structured programming, it will almost allow a number of other strategies as well. This is in contrast to Pascal, which sacrifices everything which is not consistent with top-down development in order to achieve a magnificent environment, provided one plays by its rules.

It is time for some examples. To begin with, consider the compilation process itself. Practically every book I have ever read about compiling portrays the process as a pipeline - that is a number of separate processes filtering information. Blobs of information flow into one end of each process, and new blobs, dependant solely upon the input, come out the other. These may be transformed again, until eventually executable code falls out of the end of the pipe. Graphics systems operate the same way. Now, all these books offer programs (most very well-written within the confines of the chosen language), but most feel that the program is not explanatory enough without diagrams, such as those in Figure 1.

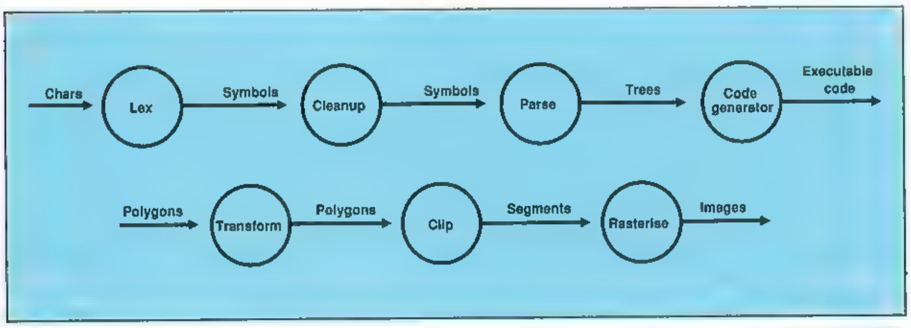


Figure 1 - An example of Pipelines

They're right - the diagram is a much better explanation of the process than a structured program. In fact, in order to get the program to work at all, a number of fiddles have to be made, which are wholly inconsistent with the structured model. They include, but are not limited to:

1. The types of all the objects flowing between the processes must be known to all the processes. In fact, each process needs to know only the types which enter and the types that leave, but this can't be expressed in a structured form.

2. While (for instance) Pascal or C have absolutely no qualms about unreading a record from a file, the idea of a stream (of which a file is an example) is not provided, because it introduces a form of side-effect. This means that a special unread procedure must be assembled for each type which must be unread. Furthermore, a variable must be declared to remember the unread item (or lack of it), and this must be available to several procedures within the process. In short, these look-ahead variables must be defined at, or very close to, the global level - short-circuiting exactly the structure which the language has been trying to enforce.

Pipelines are a special case of a much more general structure; the form is now known as Yourdon design. Ed Yourdon (a systems analyst) developed a notation which (he claimed) allowed one to understand the flow of information around a company (for example), and then translate that into a program relatively easily. An example is

dBase III Plus, FoxBASE & Clipper applications running on VAX & UNIX?

STAND 141

SOFTWARE TOOLS

Try our 30-day software licence
In an hour or two you'll be converted

Can Recital really be that easy?

The answer is yes. And to prove it, we invite you to send for our full-featured 30-day software licence for VAX/VMS™ and UNIX systems. Recital is so compatible and easy to use, you'll have your dBase™ applications up and running in Recital in about an hour. Without any additional programming or modifications.

Then take the rest of the month to discover the richness and power of this 4GL system. Recital includes tools like a report writer, screen painter, transparent access to RMS files, an integral data dictionary, popup calendars, calculators, pick lists, note pads and much more. All on a VT-type terminal!

Your 30-day license will include the full Recital system, documentation and hotline support. How can you lose?

Order today by calling one of the numbers below. Or write to Recital Corporation.

Recital Corporation Limited
South Bank Technopark
90 London Road
London SE1 6LN, UK
Tel 071-401 2727
Fax 071-633 9617

Recital
CORPORATION

Recital Corporation Inc
85 Constitution Lane
Danvers
MA 01923, USA
Tel (508) 750 1066
Fax (508) 750 8097

CIRCLE NO. 776

shown in Figure 2. The notation does have a problem in that it has very precise rules about what can go where and when - the arcs between the processes are strongly typed in a hierarchical fashion. Once a consistent drawing has been made, programming is a trivial process, although the program will usually be incomprehensible without the original drawings.

Alternatives to text

I have no complaint with Yourdon notation - indeed I think it was a very important invention. However, a number of things seem obvious to me which have never been brought to light. First, in spite of being almost totally graphical, Yourdon is a language - it has structure, type, scope, and all the other accoutrements of real languages. Second, it is not a Chomsky-type grammar, so there isn't a simple way of parsing that one can copy out of a book; nevertheless since it has precise rules, it must be compilable. Third, because the process of programming a Yourdon specification is largely mechanical, it follows that the language is unambiguous. Finally, since the programs produced in this way are incomprehensible without the original charts, it must be imposing a structure on the program which is different from that in the chosen language.

To review, here we have a structured, unambiguous, compilable language, which has about 20 years of use behind it, and is tailored towards a particular form of structure (which, incidentally, is closer to the way normal people think than top-down structure), and is so simple that valid programs can be developed away from the computer. This must be the most significant development in computer languages ever! So why is nobody compiling it?

The limits of CASE..

This, I suppose, is where CASE comes in. In fact, CASE offers the same kind of computer support as computer-aided design does - little (if any) help with design, but fairly sophisticated drawing management. Every example of CASE which I have seen so far has been the equivalent of drawing management on an established methodology. Furthermore, the methodologies which have been chosen as the bases of CASE tools have tended to be text-based disciplines which are more amenable to the computer than to the designer. One atrocious example of this is the Vienna Development Method.

VDM works like this. One defines a list of processes, each of which has certain aspects defined:

- The range of its inputs.
- The range of its outputs.
- What the process does.

Look familiar? It wouldn't in VDM! The language is designed so that a specification can be proved mathematically never to fail, and consequently for proofs to be any use the project must be specified to a better precision than even a programming language defines. While analysts were complaining that programming was too difficult even with simplified, high-level languages, here is a language which claims to solve all these problems by being more complicated!

Although VDM is a particularly insane solution to the problem of specifying programs, it is by no means unique. No, in order for abstractions and metaphors to be any use, either in specifying or writing programs, we need terms that reflect accurately the way people think. We have a contender for this: object-oriented programming.

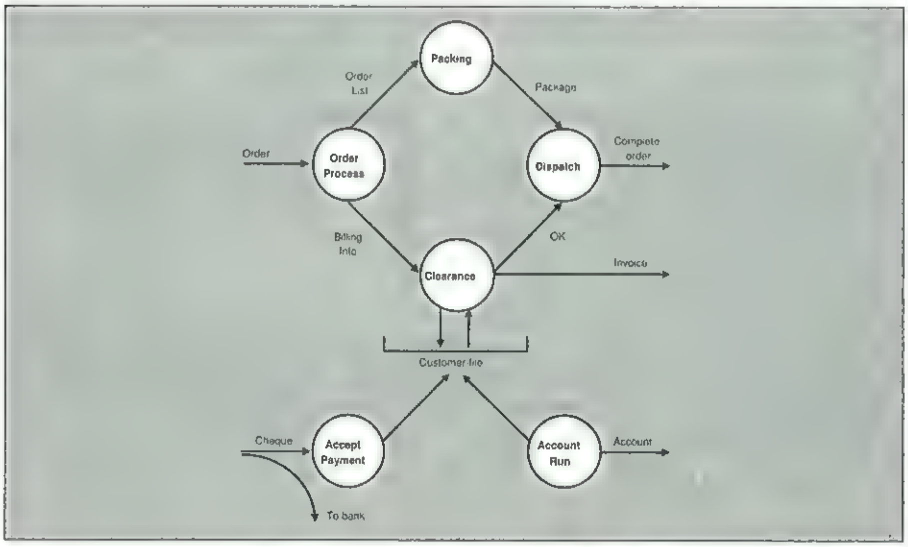


Figure 2 - An order control system expressed in Yourdon

... and OOPs

For those people who have been living in a cave for the last five years, object-oriented programming applies a totally different structure to a program than normal, declarative languages do. Instead of variables being passive repositories of data, which are acted upon by aggressive and impatient procedures, data is contained in objects, each of which knows how to do something. These objects communicate with each other by sending messages to each other, and receiving polite replies. (Object-oriented aficionados love using such anthropomorphic terms). For example, '2' would be an object, and it would know how to add itself to another, similar object, how to turn itself into a 'real', should that be required, and how to print itself on the current output stream. Objects can be declared to descend from other objects, inheriting their methods, so for instance integers and cardinals would use the same 'add' method, but it would be overridden for reals.

OOP has been around for many years, largely ignored. Suddenly it has leapt in popularity. We are being exhorted to convert completely to object-oriented development, casting aside 20 years of accumulated wisdom in the process. Is OOP really that good?

Well... yes and no. Object-orientation lends itself very well to the graphical environments I mentioned earlier, because that is exactly the metaphor windows systems use. It is no accident that Smalltalk (the granddaddy of OOPs languages) and the Mac desktop appeared in the same lab of the same company at the same time. This metaphor, though, is of desktop soup, with objects floating around in it. Anything can send any message to anything else, and there is no censorship, control, or even a system to find out where a message came from (because it is important to the image that an object not know where a message comes from). There is no concept of actively hiding information.

There is another serious problem with the OOP metaphor. Each object defines its own methods for doing things irrespective of all the other objects. Thus processing words is quite different to processing peas, and the classes 'Peas' and 'words' will both define their own methods called 'process'. However a farmer, a nutritionist and a packing clerk will all have their own idea of what it means to 'process' peas, and people of these three classes should additionally be able to define their own methods to implement these ideas. As it is, only one object

can take part in understanding a message, and in this case it would ultimately be the class 'peas'. Furthermore, there should be some concept of censorship; a secretary should not be allowed to instruct the nutritionist to process anything. In real life (which, judging by the terminology, OOPs are trying to emulate), there are validation procedures for handling messages, whether between computers in a bank, or between employees in a company. A concept of hiding information, central to Pascal and crucial in top-down design, is entirely absent from OOPs. In large projects, this leads to a feeling of anarchy, as all the classes in a system compete to decide who understands a message.

Now, I'm not saying that OOPs is a bad thing. Far from it - I think OOPs will probably point the way towards the next generation of languages - but it is not a complete solution to the software crisis of the '90s, and will, more likely, contribute to it in the short term. OOPs will show its strength in the evolution of user interface design, not just in windows, and will have an important role to play in natural language understanding. However, to claim it is in some way better than procedural (and other) languages is, I believe, throwing out the baby with the bath water.

This is one of the motivations behind the object extensions to classical languages, notably C++. There are both advantages and disadvantages to this approach. On the one hand, it gives a much more graceful introduction to appropriate design techniques, and it allows programming techniques to be mixed in just the way I am asking for. On the other hand I feel that removing the comprehensive inheritance mechanism of pure object languages (such as Smalltalk and Actor) is rather missing the point of object orientation, and there will also be a corresponding adulteration of the concept of a procedure.

A larger view

Any attempt to produce a generally applicable language must fail. People write programs by understanding a problem using metaphors (for which the problem's own jargon is the notation), and then translating that problem into notation the machine can understand. If the machine notation reflects accurately the terms used in the statement of the problem, then the program is the statement of the problem. If not, the program is a solution.

This distinction is crucial for two reasons; productivity and maintenance. Consider a cybernetic system, controlling the electronic suspension on a car. It is relatively easy to define a suite of equations which describe the movement of the car, but we want to solve those equations in reverse, and in real time. It is possible to produce a solution to this problem, by working out the relevant equations and writing a program to do the appropriate sums. The reliability of the resultant system is dependant upon the original equations (pretty good), and the calculated solutions of them (dodgy). It is safe to assume that the computer will not make mistakes in the arithmetic. Subsequently, due to engineering developments, it becomes possible to adjust the hardness of the suspension from moment to moment, and we wish to incorporate this into our control system. First, the solutions must be modified to account for the changed geometry of the suspension. Second, the system must take the decision how hard to make the suspension, presumably by trying alternatives and selecting the best.

Although this is a complex example, it graphically illustrates the points I want to make. This is a relatively simple application in terms of the problem space. The automotive engineer could state the problem using differential equations on less than one piece of paper. Furthermore, the problem, expressed as it is in mathematics, allows very little room for interpretation. The process of creating a program, however, is fraught with difficulty. When the update is required, a single statement is enough to define the change to the problem - but most of the previous solution will have to be thrown away.

The solution is obvious, if prohibitively expensive. If the problem itself were given to the computer, and the compiler were responsible for finding the solution, the change to the program would constitute a single line. The problems of maintenance, which in this case are enormous, would simply disappear.

The fact is that finding solutions to equations, which is a largely mechanical process full of detail, is exactly the kind of thing computers are good at. In a simpler form, t his is what all compilers do anyway. A good system of compiler would always allow the problem (or, if necessary, the solution to it) to be expressed in the problems' own terms. The difficulties of writing and maintaining programs all stem from the fact that a translation phase is taking place away from the computer and away from the application, and that it is being performed by a person (the programmer) who is not expert in either, but is more a kind of middleman.

Multiple languages

Although this ultimate aim is impractical (that of building programming languages whose reserved words comprise the jargon of all human knowledge), it is possible to come much closer to this ideal than is being achieved at the moment. All quantifiable knowledge (which, it is reasonable to assume, is the domain of computer solutions) is expressed using a very few metaphors. They include:

- The Process (such as in Pascal and C).
- The Object (Smalltalk, Actor, ModSim).
- Mathematics (APL, and FORTRAN if you must).
- Arithmetic (COBOL).
- The Dataflow (Yourdon and Lucid).
- The Function (WAFL, Miranda, and others).
- The Fact (Prolog).
- The String (AWK and Brief).
- The Great-big-thing-that-could-mean-anything (Lisp, dBASE, Excel etc).

I'm sure there are others, but I can't think of many more important ones. As you can see, all these metaphors have been approached at one time or another in different languages. The problem is, each language addresses one metaphor to the detriment of all others, xenophobically defending it against all alternatives. Connections into other languages have been added, if at all, as an afterthought, and even if the languages could coöperate, linkers are not adequate to handle the kind of connectivity that a fully integrated program demands. Herein lies the problem; there is no task (except synthetic tasks designed to advertise a language) which falls exclusively into one domain. The software developer has to select not the language which is the best for the job in hand, but the language which is most appropriate for the crux of the job, and then massage the rest of the job to fit the chosen language.

If a solution is to be found to the next software crisis, (and, rest assured, it will come), it will be found not in more complex and clever languages, but in linkage.

EXE

Jules May is a freelance programmer specialising in graphics and HCI. He also runs Jules Computer Ltd, selling graphics systems and productivity tools. He can be contacted on 0707 44185 or on CIX as Jules.

Watch out for Jules' regular .EXE column, which begins next month.

Babbage's Analytical Engine: A Programmer's View

This year sees the bicentenary of the birth of Charles Babbage, the pioneer of the computer. Martin Campbell-Kelly looks at the programming ideas behind his Analytical Engine, the first design for an automatic digital computer.

In 1842 the Astronomer-Royal Sir George Biddell Airy was asked by the Chancellor of the Exchequer whether or not the Government should spend any further money on Charles Babbage's calculating engine. His opinion was 'that it was worthless'. Today Airy is largely forgotten, but in 1991 Britain is celebrating the bicentenary of the birth of Charles Babbage. In March, a commemorative postage stamp was issued by the British Post Office; his Difference Engine No. 2 is being reconstructed by the Science Museum; and there will be two major conferences on Babbage during the summer. This article explores Babbage's crowning intellectual achievement, the Analytical Engine, from the point of view of today's programmer.

Charles Babbage

Charles Babbage was born in London on Boxing Day, 1791, the son of a wealthy banker. Although today Babbage is best known as the pioneer of the computer, he was much more than this - he was a vigorous reformer and a polymath in the nineteenth-century tradition (Figure 1).

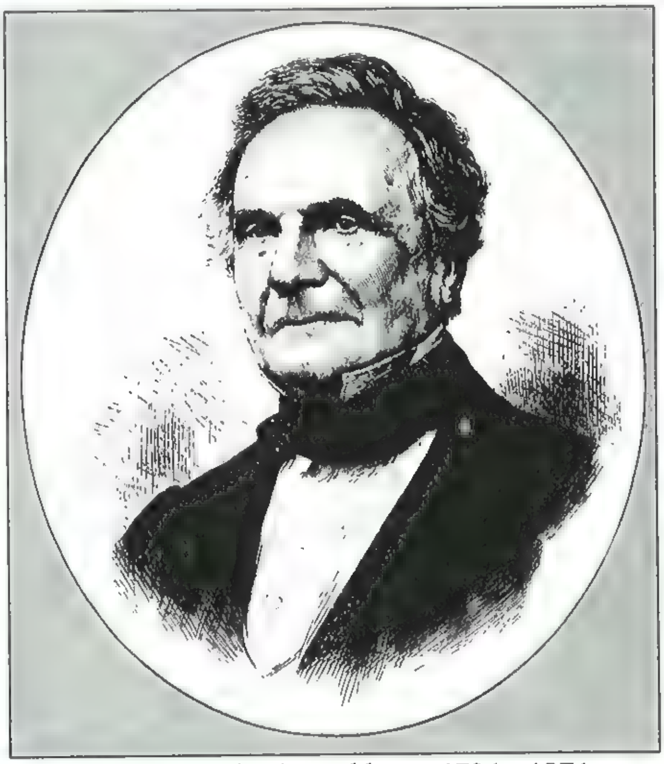


Figure 1 - Charles Babbage, 1791 - 1871

Shortly after graduating in mathematics from Cambridge University in 1812, Babbage and two of his reforming contemporaries initiated a renaissance of British mathematics by introducing the modern notation for the calculus. In 1828 he was appointed Lucasian Professor of Mathematics at the University. He played a major role in raising the status of science in Britain from being a pastime for gentlemen amateurs to being a professional pursuit. His classic book *On the Economy of Machinery and Manufactures* (1832) was the first serious treatment of the micro-economics of manufacturing. He made contributions across the spectrum of science - publishing papers on the physical sciences, statistics, technology, and geology. He made numerous inventions, including a medical opthalmoscope. He was an expert on cryptography, the railways and insurance. He once campaigned to be a Member of Parliament, and he wrote pamphlets on political and economic reform. He even wrote a semi-religious book on Natural Theology. Today, however, it is chiefly for his calculating engines that Charles Babbage is remembered.

The Difference Engine

According to Babbage's (not always reliable) autobiography, he first conceived the idea of a calculating engine while still an undergraduate at Cambridge: 'One evening I was sitting in the rooms of the Analytical Society, at Cambridge, my head leaning forward on the table in a kind of dreamy mood, with a table of logarithms lying open before me. Another member, coming into the room, and seeing me half asleep, called out, "Well, Babbage, what are you dreaming about?", to which I replied, "I am thinking that all these tables (pointing to the logarithms) might be calculated by machinery".'

It was not until 1822 that Babbage drew up concrete proposals for his Difference Engine, a machine that would calculate and print mathematical tables completely automatically. The calculations were to performed by a mathematical process known as the Method of Differences - hence the name Difference Engine.

Babbage succeeded in obtaining funds from the Government to construct the engine, arguing that it would be used to produce nautical and other tables in the national interest. Eventually he received a total of £17,000 - the equivalent of perhaps £1 million today. Although Babbage worked on the project for a decade, no machine was ever produced, except for the small prototype which is now displayed in the Science Museum, Kensington.

There were several reasons for Babbage's failure to produce a full-scale machine. Not least, there was the formidable engineering problem of producing a reliable machine of several thousand components using the crude mechanical-engineering technology of his day. Another major problem was Babbage's inability to freeze his specification, so that he constantly altered the design while the engine was being built. This, and Babbage's abrasive manner, caused a good

Security Software from Microft Technology.

MENUGEN

MENUGEN provides a password protected menu interface. The system manager has complete control of what each user is permitted to do. Access to the DOS prompt can be prevented. MENUGEN has no memory overhead.

D-LOCK

D-LOCK prevents access to the hard disk after booting from diskette.

CLAM

CLAM provides TOTAL security. It incorporates MENUGEN and D-LOCK. In addition it provides a number of other facilities, the most important of which is file encryption – the only way to provide a high level of security on a micro.

Microft Technology Limited
The Old Powerhouse, Kew Gardens Station
Kew, Surrey TW9 3PS.
Telephone: 081-948 8255

CLAM £148+VAT Single User.
MENUGEN £48+VAT Single User.
D-LOCK £38+VAT Single User.
Corporate Licences available.

CIRCLE NO. 779

deal of animosity between him and the engineer he commissioned to build the machine and, of course, with the Government which was funding the project. In 1834 the Government lost patience and withdrew its support for the Difference Engine.

In fact, in about 1834, Babbage had produced a remarkable new design which he called the Analytical Engine. The Analytical Engine was to be a program-controlled general-purpose computer, very close in spirit to the modern digital computer. By contrast, the Difference Engine was a special-purpose machine which could do nothing except produce mathematical tables. Important as table-making was, the Analytical Engine was a different order of conception altogether. It was the supreme achievement of Babbage's intellectual life, and he pressed ahead with the project despite the uncertainties of funding.

The Analytical Engine

One of the key breakthroughs that Babbage made in the Analytical Engine was to separate what we now call the processor from the memory. In fact Babbage used the terms 'mill' and 'store', in an analogy with factory organisation: thus numbers were taken from the store into the mill for arithmetic processing and the results were then sent back to the store. For his input-output units, Babbage decided to use punched cards of the type used in Jacquard Looms, which were used in the weaving industry for making tapestries. (The same basic technology can still be seen in use in old-fashioned fairground organs.) Babbage also decided to use punched cards for his programs. Thus by simply changing the program, the Analytical Engine could immediately be set to work on another calculation. All these ideas seem so self-evident today that it is difficult for us to fully comprehend what a giant intellectual leap Babbage had made.

Figure 2 shows a general plan of the Analytical Engine, dating from 1840. The mill (or processor) is the circular arrangement to the left of the drawing. The mill, which would have been about six foot in diameter, performed the four arithmetic operations of addition, subtraction, multiplication, and division. Addition and subtraction operations would have taken about three seconds, while multiplication and division would have taken about a minute. The store is shown to the right of the picture. In the store, the individual elements (or 'variables') were designated V_0, V_1, V_2, ...; only a few storage elements are shown in the drawing, but Babbage envisioned many more being provided in the finished machine. Each variable was to be held on a vertical column of 50 digit-wheels, about 10 foot high. The complete Analytical Engine would thus have been about the size and weight of a small railway locomotive.

Programming the Analytical Engine

Although Babbage wrote a good deal in his lifetime - his collected works amount to 11 volumes - he published very little indeed on the Analytical Engine. The best contemporary account we have is Ada Lovelace's *Sketch of the Analytical Engine* published in 1843. This article contained a number of 'programs' for the Analytical Engine, and many writers have assumed that these were Lovelace's own work. She has frequently been described as the world's first programmer, and the Ada programming language was named in her honour. In fact, the original computations were all devised by Babbage, and he himself described Lovelace as 'my dear and much admired interpreter'. Although Lovelace's role as a programmer has been exaggerated, this should not blind us to her achievement in the important, if less spectacular, role of interpreting the Analytical Engine for an uncomprehending world. Even today Lovelace's Sketch remains the most cited, and one of the clearest, accounts of programming the Analytical Engine.

Figure 3 shows the simplest example given in the Sketch. The problem (given in the awkward notation of the period) is to solve the pair of simultaneous equations

$$mx + ny = d$$
$$m'x + n'y = d'$$

using the formula

$$x = \frac{dn' - d'n}{n'm - nm'}$$

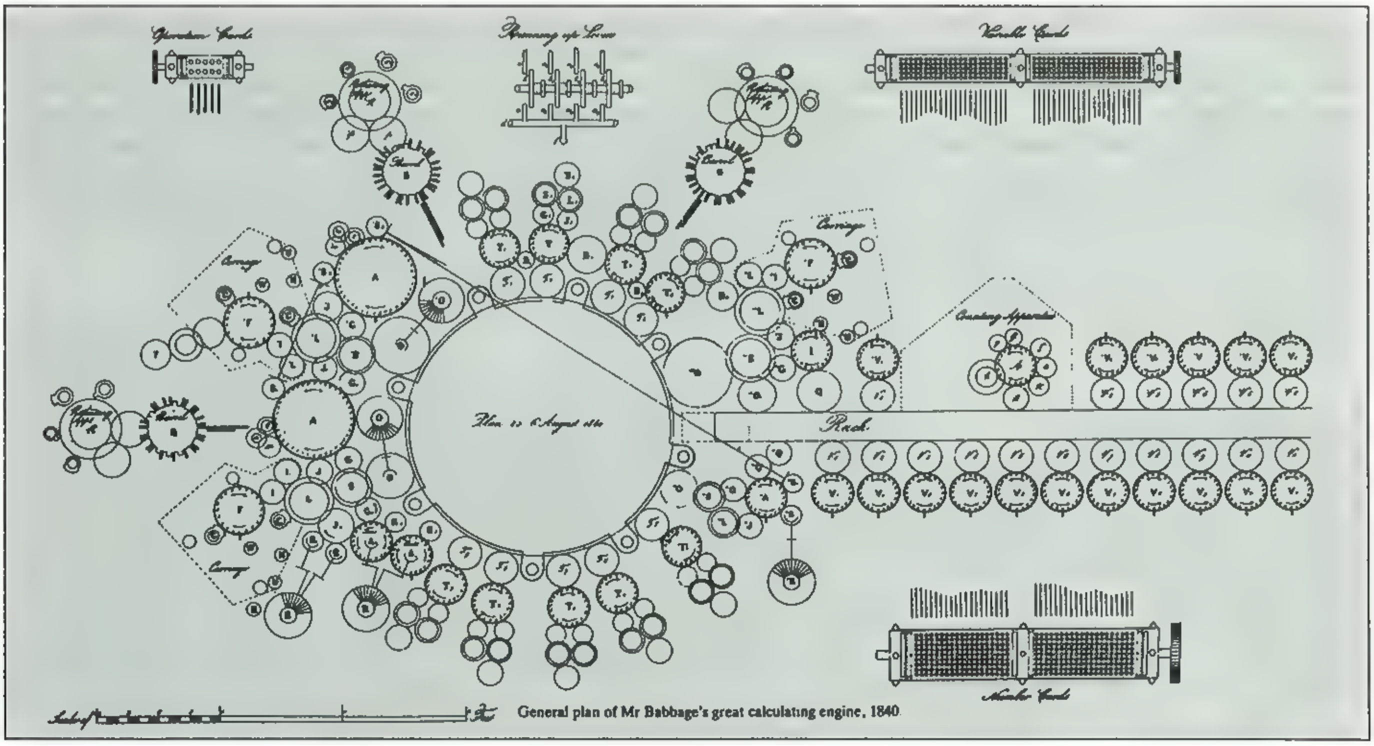


Figure 2 - General plan of the Analytical Engine, 1840

Number of the operations	Operation cards	Cards of the Variables		Progress of the operations
	Symbols indicating the nature of the operations	Columns on which operations are to be performed	Columns which receive results of operations	
1	×	$V_2 \times V_4 =$	V_8	$= dn'$
2	×	$V_5 \times V_1 =$	V_9	$= d'n$
3	×	$V_4 \times V_0 =$	V_{10}.......	$= n'm$
4	×	$V_1 \times V_3 =$	V_{11}.......	$= nm'$
5	−	$V_8 - V_9 =$	V_{12}.......	$= dn' - d'n$
6	−	$V_{10} - V_{11} =$	V_{13}.......	$= n'm - nm'$
7	÷	$\frac{V_{12}}{V_{13}} =$	V_{14}.......	$= x = \frac{dn' - d'n}{n'm - nm'}$

Figure 3 - A simple 'program' for the Analytical Engine

It is assumed that the coefficients *n, m, d* and *n', m', d'* are stored in the variables V_0, V_1, V_2, and V_3, V_4, V_5. The result, *x*, is left in V_{14}. (The computation could easily be extended to calculate *y*.)

A program for the Analytical Engine was punched onto two sets of cards: a set of operation cards and a set of variable cards. Each step of the program was rather like a three-address instruction for a modern computer, with the interesting difference that the operation and the three 'addresses' were not bound into a single entity. The execution of the program is very clear in Figure 3, which looks a little like a program walkthrough - and indeed, this is exactly what it is. In fact, Babbage did not have the concept of a program as we understand it today: that is, a static piece of code which is dynamically executed by the computer. All of Babbage's examples are essentially walkthroughs: they show how the Analytical Engine would have performed a particular task, but not how the program cards would have been punched. Thus the programs contain no explicit conditional or branch statements, or input-output operations.

Bernoulli Numbers

The previous example illustrated only straight-line code. The modern digital computer gains its power from conditional instructions, program loops and indexed addressing. To evaluate the Analytical Engine, we therefore need to consider how it would have fulfilled to these functions. This is best illustrated by the Bernoulli Numbers example, which is one of the most complicated that Babbage produced. The original diagram for the computation is shown in Figure 4, while Figure 5 shows a transliteration into the Pascal programming language. (Bernoulli Numbers, incidentally, are based on the coefficients of the series expansion of $x/(e^x - 1)$, which occurs in a number of mathematical contexts. The first few Bernoulli Numbers computed in the Babbage program are $B_1 = 1/6$, $B_3 = -1/30$, $B_5 = 1/42$ etc. The details are not important here, but a good exposition can be found in Volume 1 of DE Knuth's *Art of Computer Programming.*)

The Analytical Engine performed conditional operations by detecting a change in the sign of a number. The Jacquard cards that represented the program were all strung together, so that detecting a change of sign - 'running up' as Babbage called it - enabled subsequent operations to be taken from a different part of the program. The effect of this would have been similar to a conditional branch instruction on a modern computer, but Babbage certainly did not have the modern notion of a goto-statement. Indeed, although there are a number of conditional operations in the Bernoulli computation, they are all implicit.

Again, the Bernoulli example makes an implicit use of a program loop. In the diagram for the computation this is indicated by the statement 'Here follows a repetition of Operations thirteen to twenty-three'. The exact mechanism by which looping was to be achieved is obscure, but essentially it would have involved 'backing' the program cards, so that the cards that constituted the program loop would be obeyed repeatedly. A 'Card Counting Apparatus', which was set up with an integer value that automatically decremented each cycle of the loop, would determine the number of iterations.

The ability of the modern computer to perform vector processing and indexed addressing is the key feature that distinguishes it from its forerunners of the 1930s and 1940s. It was this feature that enabled it to perform matrix calculations (eg for partial differential equations in weapons design or weather forecasting), as well as sorting and retrieval operations. It is, therefore, particularly interesting to see how far Babbage came towards realising this concept.

The Bernoulli example shows the use of a vector to store the Bernoulli numbers B_1, B_3, B_5, ...; but once again the actual mechanism is obscure, and it seems unlikely that Babbage resolved all the technical issues. For more evidence on this, Babbage scholars have recently examined his papers in the Science Museum, where there are some two-dozen programs. One series of programs shows his attempts to generalise solving a set of simultaneous equations by the method of Gaussian Elimination. In these programs Babbage is clearly attempting to find a way of structuring the data in the store so that it is possible to solve a linear system of any order. The general strategy used was to take advantage of the independence of the operation and variable cards, so that each time round a loop the arithmetic process would be performed on the successive elements of a vector. But the results were decidedly complicated and messy, and Babbage never fully generalised the program.

In the modern computer, problems of this type are programmed by regarding a memory address as a variable numeric quantity which can be manipulated by the program (typically via an index register). This enables memory to be accessed in a systematic way for processing vectors and lists. Babbage never came up with what we now call the variable-address concept, and so he failed to take the final step towards a machine with the power of the modern digital computer.

In order to transliterate the Bernoulli Numbers example into a modern programming language (Figure 5), all the conditional and branch statements have been made explicit, along with the data-initialisation and output statements, and modern array accessing has been used. No change has been made to the logic of the computation, however, and the resulting program *almost* correctly computes the Bernoulli numbers. In fact, one statement needed to be corrected - operation number 24 had to be changed from an addition to a subtraction - and then the results were exactly correct. In effect this is a program bug; and very probably one could claim some kind of record for this being the longest time ever taken to discover a bug!

Number of Operation	Nature of Operation	Variables acted upon	Variables receiving results	Indication of change in the value on any Variable	Statement of Results	Data ${}^{1}V_{1}$ 0001 [1]	Data ${}^{1}V_{2}$ 0002 [2]	Data ${}^{1}V_{3}$ 0004 [n]	${}^{0}V_{4}$ 0000	${}^{0}V_{5}$ 0000	${}^{0}V_{6}$ 0000
1	×	${}^{1}V_{2} \times {}^{1}V_{3}$	${}^{1}V_{4}, {}^{1}V_{5}, {}^{1}V_{6}$	$\left\{ {}^{1}V_{2} = {}^{1}V_{2} \atop {}^{1}V_{3} = {}^{1}V_{3} \right\}$	$= 2n$	...	2	n	2n	2n	:
2	−	${}^{1}V_{4} - {}^{1}V_{1}$	${}^{2}V_{4}$	$\left\{ {}^{1}V_{4} = {}^{2}V_{4} \atop {}^{1}V_{1} = {}^{1}V_{1} \right\}$	$= 2n-1$	1	...	...	2n−		
3	+	${}^{1}V_{5} + {}^{1}V_{1}$	${}^{2}V_{5}$	$\left\{ {}^{1}V_{5} = {}^{2}V_{5} \atop {}^{1}V_{1} = {}^{1}V_{1} \right\}$	$= 2n+1$	1	...	...	...	2n+	
4	÷	${}^{2}V_{5} \div {}^{2}V_{4}$	${}^{1}V_{11}$	$\left\{ {}^{2}V_{5} = {}^{0}V_{5} \atop {}^{2}V_{4} = {}^{0}V_{4} \right\}$	$= \frac{2n-1}{2n+1}$	...	...	...	0	0	...
5	÷	${}^{1}V_{11} \div {}^{1}V_{2}$	${}^{2}V_{11}$	$\left\{ {}^{1}V_{11} = {}^{2}V_{11} \atop {}^{1}V_{2} = {}^{1}V_{2} \right\}$	$= \frac{1}{2} \cdot \frac{2n-1}{2n+1}$	...	2	...	...	...	...
6	−	${}^{0}V_{13} - {}^{2}V_{11}$	${}^{1}V_{13}$	$\left\{ {}^{2}V_{11} = {}^{0}V_{11} \atop {}^{0}V_{13} = {}^{1}V_{13} \right\}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$	...	...	...	...	...	...
7	−	${}^{1}V_{3} - {}^{1}V_{1}$	${}^{1}V_{10}$	$\left\{ {}^{1}V_{3} = {}^{1}V_{3} \atop {}^{1}V_{1} = {}^{1}V_{1} \right\}$	$= n - 1 (=3)$	1	...	n	...	...	...
8	+	${}^{1}V_{2} + {}^{0}V_{7}$	${}^{1}V_{7}$	$\left\{ {}^{1}V_{2} = {}^{1}V_{2} \atop {}^{0}V_{7} = {}^{1}V_{7} \right\}$	$= 2 + 0 = 2$	...	2	...	...	...	...
9	÷	${}^{1}V_{6} \div {}^{1}V_{7}$	${}^{3}V_{11}$	$\left\{ {}^{1}V_{6} = {}^{1}V_{6} \atop {}^{0}V_{11} = {}^{3}V_{11} \right\}$	$= \frac{2n}{2} = A_1$	...	...	...	...	...	2
10	×	${}^{1}V_{21} \times {}^{3}V_{11}$	${}^{1}V_{12}$	$\left\{ {}^{1}V_{21} = {}^{1}V_{21} \atop {}^{3}V_{11} = {}^{3}V_{11} \right\}$	$= B_1 \cdot \frac{2n}{2} = B_1 A_1$	...	...	...	...	...	...
11	+	${}^{1}V_{12} + {}^{1}V_{13}$	${}^{2}V_{13}$	$\left\{ {}^{1}V_{12} = {}^{0}V_{12} \atop {}^{1}V_{13} = {}^{2}V_{13} \right\}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}$	...	...	...	...	...	...
12	−	${}^{1}V_{10} - {}^{1}V_{1}$	${}^{2}V_{10}$	$\left\{ {}^{1}V_{10} = {}^{2}V_{10} \atop {}^{1}V_{1} = {}^{1}V_{1} \right\}$	$= n - 2 (=2)$	1	...	...	...	...	...
13	−	${}^{1}V_{6} - {}^{1}V_{1}$	${}^{2}V_{6}$	$\left\{ {}^{1}V_{6} = {}^{2}V_{6} \atop {}^{1}V_{1} = {}^{1}V_{1} \right\}$	$= 2n - 1$	1	...	...	...	...	2n
14	+	${}^{1}V_{1} + {}^{1}V_{7}$	${}^{2}V_{7}$	$\left\{ {}^{1}V_{1} = {}^{1}V_{1} \atop {}^{1}V_{7} = {}^{2}V_{7} \right\}$	$= 2 + 1 = 3$	1	...	...	...	...	...
15	÷	${}^{2}V_{6} \div {}^{2}V_{7}$	${}^{1}V_{8}$	$\left\{ {}^{2}V_{6} = {}^{2}V_{6} \atop {}^{2}V_{7} = {}^{2}V_{7} \right\}$	$= \frac{2n-1}{3}$	...	...	...	...	...	2n−1
16	×	${}^{1}V_{8} \times {}^{3}V_{11}$	${}^{4}V_{11}$	$\left\{ {}^{1}V_{8} = {}^{0}V_{8} \atop {}^{3}V_{11} = {}^{4}V_{11} \right\}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3}$	...	...	...	...	...	
17	−	${}^{2}V_{6} - {}^{1}V_{1}$	${}^{3}V_{6}$	$\left\{ {}^{2}V_{6} = {}^{3}V_{6} \atop {}^{1}V_{1} = {}^{1}V_{1} \right\}$	$= 2n - 2$	1	...	...	...	...	2
18	+	${}^{1}V_{1} + {}^{2}V_{7}$	${}^{3}V_{7}$	$\left\{ {}^{2}V_{7} = {}^{3}V_{7} \atop {}^{1}V_{1} = {}^{1}V_{1} \right\}$	$= 3 + 1 = 4$	1	...	...	...	...	...
19	÷	${}^{3}V_{6} \div {}^{3}V_{7}$	${}^{1}V_{9}$	$\left\{ {}^{3}V_{6} = {}^{3}V_{6} \atop {}^{3}V_{7} = {}^{3}V_{7} \right\}$	$= \frac{2n-2}{4}$	...	...	...	...	...	2n−
20	×	${}^{1}V_{9} \times {}^{4}V_{11}$	${}^{5}V_{11}$	$\left\{ {}^{1}V_{9} = {}^{0}V_{9} \atop {}^{4}V_{11} = {}^{5}V_{11} \right\}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{4} = A_3$	...	...	...	...	...	
21	×	${}^{1}V_{22} \times {}^{5}V_{11}$	${}^{0}V_{12}$	$\left\{ {}^{1}V_{22} = {}^{1}V_{22} \atop {}^{0}V_{12} = {}^{2}V_{12} \right\}$	$= B_3 \cdot \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{3} = B_3 A_3$	...	...	...	...	...	...
22	+	${}^{2}V_{12} + {}^{2}V_{13}$	${}^{3}V_{13}$	$\left\{ {}^{2}V_{12} = {}^{0}V_{12} \atop {}^{2}V_{13} = {}^{3}V_{13} \right\}$	$= A_0 + B_1 A_1 + B_3 A_3$	...	...	... ...	...	...	...
23	−	${}^{2}V_{10} - {}^{1}V_{1}$	${}^{3}V_{10}$	$\left\{ {}^{2}V_{10} = {}^{3}V_{10} \atop {}^{1}V_{1} = {}^{1}V_{1} \right\}$	$= n - 3 (=1)$	1	...	...	...	...	...
					Here follows a repetition of Operations thirteen to twenty-three						
24	+	${}^{4}V_{13} + {}^{0}V_{24}$	${}^{1}V_{24}$	$\left\{ {}^{4}V_{13} = {}^{0}V_{13} \atop {}^{0}V_{24} = {}^{1}V_{24} \right\}$	$= B_7$	...	...	...	...	...	..
25	+	${}^{1}V_{1} + {}^{1}V_{3}$	${}^{1}V_{3}$	$\left\{ {}^{1}V_{1} = {}^{1}V_{1} \atop {}^{1}V_{3} = {}^{1}V_{3} \right\}$ $\left\{ {}^{3}V_{6} = {}^{0}V_{6} \atop {}^{3}V_{7} = {}^{0}V_{7} \right\}$	$= n + 1 = 4 + 1 = 5$ by a Variable-card. by a Variable-card.	1	...	n+1	...	...	0

Figure 4 - The Bernoulli Numbers computation (part)

```
program Bernoulli (output);
  label 1, 13,24;
  var q: 0..99;
      V: array[0..99] of real;

begin
  V[1] := 1; V[2] := 2; V[3] := 1;

  { operations 1 - 7 }
  1:
    q := 21;
    V[4] := V[2] * V[3];
    V[5] := V[4];
    V[6] := V[4];
    V[4] := V[4] - 1;
    V[5] := V[5] + V[1];
    V[11] := V[4] / V[5];
    V[11] := V[11] / V[2];
    V[13] := -V[11];
    V[10] := V[3] - V[1];
    if V[10] = 0 then goto 24;

  { operations 8 - 12 }
    q := 22;
    V[7] := V[2];
    V[11] := V[6] / V[7];
    V[12] := V[21] * V[11];
    V[13] := V[12] + V[13];
    V[10] := V[10] - V[1];
    if V[10] = 0 then goto 24;

  { operations 13-23 }
  13:
    V[6] := V[6] - V[1];
    V[7] := V[7] + V[1];
    V[8] := V[6] / V[7];
    V[11] := V[8] * V[11];
    V[6] := V[6] - V[1];
    V[7] := V[7] + V[1];
    V[9] := V[6] / V[7];
    V[11] := V[9] * V[11];
    V[12] := V[q] * V[11];
    q := q + 1;
    V[13] := V[12] + V[13];
    V[10] := V[10] - V[1];
    if V[10] <> 0 then goto 13;

  { operations 24-25 }
  24:
    V[q] := -V[13];
    writeln('B[', 2*(q-21)+1:2, '] =',
V[q]:8:4);
    V[3] := V[3] + V[1];
    goto 1;
end.
```

Output:

```
B[ 1] =  0.1667   [ie 1/6 ]
B[ 3] = -0.0333   [ie -1/30 ]
B[ 5] =  0.0238   [ie 1/42 ]
B[ 7] = -0.0333   [ie -1/30 ]
B[ 9] =  0.0758   [ie 5/66 ]
B[11] = -0.2531   [ie -691/2730 ]
etc
```

Figure 5 - The Bernoulli Numbers program in Pascal

Assessment

The Government formally withdrew support for the calculating engines in 1842, and after that Babbage gave up all hope of ever constructing the Analytical Engine. Although he continued to refine his designs to his last years, this was essentially a hobby for his old age. He died in 1871 at the age of 79. His son Henry completed an experimental fragment of the Analytical Engine in about 1908, which is preserved in the Science Museum; but after that Babbage's computing work was neglected for half a century until it was seriously taken up by the present generation of computer historians.

Consequently, the modern digital computer - often called the *von Neumann* architecture after one of its principal inventors - was developed in complete ignorance of Babbage's achievements. Perhaps the most remarkable thing about the Analytical Engine is its extraordinary likeness to the modern computer, in spite of being designed independently, in another century, and using an entirely different technology. As Allan Bromley, a noted historian of computing, has put it: 'Indeed, I am bothered that the Analytical Engine is too much like a modern computer. Do we infer that a computer can only be built in one sort of way? Or have we allowed ourselves to be backed into a corner in using only one computational style?' This is an issue that is very germane to current attempts to break away from the von Neumann-style of computing.

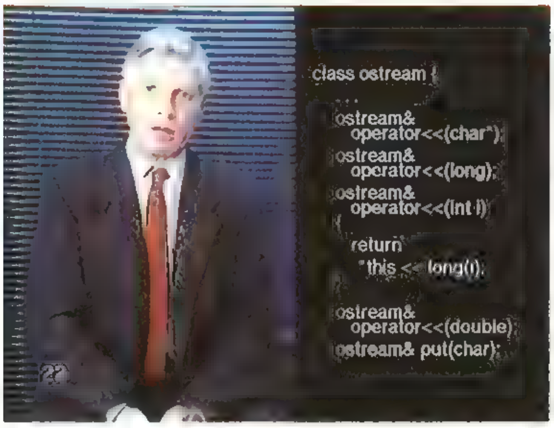
How then should we remember Babbage in his bicentennial year? Clearly, if it was not for the importance of the computer in the late twentieth century, we would probably not be remembering him at all. And yet the modern digital computer owes nothing to Babbage's Analytical Engine. Indeed, it is possible to argue that Babbage's failure to complete his machine acted as a negative influence - creating a climate of failure that discouraged people from attempting to construct an automatic calculating machine for several decades. Be this as it may, it cannot take away from him the monumental achievement of conceiving a computer a hundred years before electronics made such machines a practical reality.

EXE

Martin Campbell-Kelly lectures in computer history at the University of Warwick. He is the editor of the Works of Charles Babbage, published by Pickering & Chatto in 1989.

VIDEO COURSE

Learn C++



NO RUSH!
NO TRAVEL!
NO HOTELS!
ONLY £299.95

Learn C++ Now!
The great exodus of programmers from C to C++ has begun! Since C++ builds on C, it's the easiest OOP language to learn. That's why it's called "the language of the 90's".

Why the rush?
PRODUCTIVITY! Yes, C++ programmers can write programs in less time requiring less maintenance. Large projects become much easier to manage.

Unfortunately, learning C++ can be very costly. Classroom instruction is expensive even without the travel and hotel costs. Of course, not learning will cost you even more in the long run. Now there is an alternative!

The top C++ video tutorial at the lowest possible price.
The C++ video tutorial from Zortech is the ultimate C++ training tool for work or home at only £299.95. It comes on six VHS video tapes containing 32 clear, extensive tutorials.

Used in conjunction with the concise workbook and tutorial disk, you will find everything you need for fast-track C++ tuition.

The course is generic (i.e. compiler and hardware independent) and is available with or without the award winning Zortech C++ Compiler for MS-DOS and OS/2.

Start writing C++ code within a week.
As a C programmer, you will start producing C++ code within a week of concentrated use of this course. Alternatively, spend just an hour a day watching the video and working through the suggested exercises to learn C++ in only six weeks!

The Leader in video tutorials.
If you don't already know C, you can join tens of thousands of programmers world-wide who have learnt C with the Zortech Complete C Video Course, described as:

"An excellent bargain ... ... I heartily recommend"
Gary Ray, PC WEEK

Save your company thousands of £££'s.
One programmer can train for only £299.95, but you can train ten programmers for just:

C++ Course	£299.95
9 Extra Workbooks	£179.55
Total 10 students	£479.50

Yes! Only £47.95 each! (With all the FREE refresher courses you need!)

Now, Zortech with its new C++ Video Tutorial has refined the art of video tuition and presentation even further for the 1990's.

IN EUROPE CALL:

44- 1-316-7777

Only £299.95 complete
- *Six Videos with 32 lessons*
- *256 page workbook*
- *Tutorial disk*
- *Compiler & hardware independent*
- *NTSC or PAL format*
- *Tax deductible*

Don't delay, order now!
Just mail the coupon or call the order hotline for same day shipment.

USA: Zortech Inc.,
1165 Massachusetts Avenue,
ARLINGTON, MA02174
Voice: 617-646-6703
Fax: 617-643-7969

EUROPE: Zortech Ltd.,
106-108 Powis Street, LONDON
SE18 6LU
Voice: 44- 1-316-7777
Fax: 44- 1-316-4138

ORDER FORM
Please rush me these items:

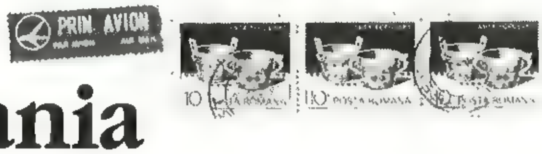
Qty	Description	Price
___	C++ Video Tutorial	£299.95
___	Extra C++ Workbooks	£. 19.95
___	Zortech C++ Compiler	£129.95
___	Developer's Edition	£299.95

Please add £5 + VAT postage in UK
All UK orders please add 15% VAT
International shipping charged at cost

Name ____________
Company ____________
Address ____________
____
____
Phone ____________
MC,VISA or CHQ ________ Expiry ____
Card No. ____________

EXE 5/91

CIRCLE NO. 784



Hard Times in Romania

All countries east of the old Iron Curtain have fallen behind in the technology race, but Romania, ruled by a particularly tyrannical dictatorship, has perhaps suffered more than most. Doru Turturea and Dan Somnea explain.

Before discussing Romanian IT - we say 'informatics' - there are a few things we ought to get straight:

- Romania has the smallest and most elderly selection of mainframes, minis and microcomputers in Europe.
- There is an acute shortage of systems analysts.
- The number of good programmers is a very small fraction of the population of programmers.
- Because of our mainframe hardware, we need programmers who know how to work using punched cards for input. Needless to say, the majority of the programmers are more interested in the interactive systems available on our few minis.

A brief history. Between the years 1967 and 1975, Romania enjoyed a short period of comparative openness. Our Government even co-operated with the IBM corporation. But from 1975, the country was more and more marginalised by the dictatorship, and this was reflected in the way informatics was taught. The gap between our computer industry and that in Western Europe, already large, got huge. Universities were starved of equipment. Although they backed industry by providing informatics graduates, they could not supply the necessary skills. The dictatorship used the number of universities which offered this informatics degree as a symbol of success. There were a few good, isolated experiments, especially in the 1980s; for example, some classes of children were taught BASIC on 8-bit personal computers (Sinclair Spectrum compatibles). But today, we still have insufficient numbers of trained teachers of informatics.

However, the current government is very keen on the idea of building an information-based society. We hope to get help from the universities and colleges of other European countries.

Hardware

Let's have a quick look at the state of our computer industry.

Our mainframes: the Felix C256, C512/1024 are all based on the batch-processing philosophy. The old C256 was modelled on a French design (CII-IRIS 50) dating from the 1970s. This was a symptom of political folly: our experts urged the Government to foster our relationship with IBM, but instead it ignored technical advice and imposed the French system. Later on, when our Eastern block neighbours started to build their own mainframes (based on the so-called 'Edinii project'), their equipment was IBM 370 compatible. Romanian mainframes remained isolated, compatible with nothing.

The operating system of these machines permits some multiprocessing, with jobs being spooled to disk. Both synchronous and asynchronous communications are supported, allowing the connection of VDU terminals. Manufacture of these machines has now practically ceased. Just before the Revolution, our factory began to produce a new, more compact generation of mainframes called the Felix 5000 series.

We have done better with minicomputers. When our hardware experts started to design and manufacture minis, they chose to follow DEC. The architectures of the minis were compatible with the DEC PDP families including, latterly, the VAX 780. At the same time, our industry has started to produce Romanian-designed VT-compatible VDUs (VT52s, VT100s and VT200s).

In the 1980s, we began to produce small quantities of micros: CP/M-80 and Sinclair Z80 compatible machines. Towards the end of the decade, we even managed to put together a few PCs and XTs - the IBM-PC compatible design coming from our Polytechnic Institute of Bucharest. The low volume of products was caused by a shortage of parts. Our mini and micro hardware was generally designed three to eight years before it trickled into production. This was because, before the Revolution, it was impossible to import the LSI, MSI and VLSI chips that we needed from the West.

Software

The lack of good IDEs or software tools has had a decisive negative influence on the Romanian software development community. Our programmers have learned old-fashioned programming languages (COBOL and FORTRAN on mainframes, FORTRAN 77, COBOL and Pascal on minis), writing many applications. Some of our mainframe programmers have been able to use DBMS packages, like Socrate and Oracle. The story for our CP/M-80 micros is pretty similar (FORTRAN, COBOL and C), although these machines are much more used for packages then for programming. The most popular applications were dBASE and Wordstar. Because there were so few IBM PC compatibles before the Revolution, very few of us had encountered popular software such as Turbo Pascal, TopSpeed Modula-2, Symphony and so on.

At the Research Institute for Computers, where we work, we have developed the 'U' operating system (a UNIX System V 2.3-alike) which runs across various minis and micros. On the PC-level, we are now developing applications using Turbo C V2.0 to run under MS-DOS 3.3 and Windows 2.0.

As for the future, we think that the demand for dBASE, Lotus and C programs will increase dramatically, causing a shortage of PC applications programmers. We know that the 1980s was the PC era, and this decade is supposed to be the era of interdependent computing, so we hope to be able to install many Novell LANs. This should be possible, now that the COCOM embargo has been relaxed. But in the long term, especially in the academic world, we have out sights set on UNIX.

EXE

Mr Dan Somnea is Senior Analyst and assistant professor in the Cybernetics, Informatics and Statistics Department. He is a member of the Romanian Computer Science Society.

Mr Doru Turturea is a researcher in the Institute of Computing Techniques. He is the secretary of the Romanian Computer Science Society.

Programming for CD-ROM

The ISO 9660 standard for CD-ROM has come to fruition at just the right time, as software suppliers see the advantages in this high-density, low cost medium. Michael Price explains.

More and more software suppliers are turning to CD-ROM as an inexpensive distribution and storage medium. With a CD-ROM disk holding up to 650 MB of data, and volume production costs of around £2 per disk, it is easy to see the attraction. The technology is already firmly established in the Macintosh environment and it is becoming standard in the UNIX arena.

As MS-DOS and OS/2 applications grow in size, it may now be time for widespread use of CD-ROM in these systems. This has been intensified by the development of standards which mean that any CD-ROM drive can be used to access any CD-ROM disk using standard operating system facilities.

The Disk

CD-ROM (Compact Disk - Read Only Memory) disks are similar to conventional CD Audio disks. They are manufactured using the same production techniques (though with enhanced quality control) and contain inherently similar data structures.

The data on a CD disk is organised along a spiral path that is traversed at constant linear velocity. The path is divided into blocks of 2352 bytes. Each block has an absolute address measured in minutes, seconds and 75ths of a second. A time and position algorithm built into the CD-ROM drive microprocessor is used to select a specific block to begin data retrieval.

The 2352 bytes in each block are assigned to 12 synch bytes, 3 address bytes (min, sec, block number), a mode byte and 2336 bytes of user data. For audio disks, this is all usable information. In the case of CD-ROM blocks, the user data is divided into 2048 bytes of actual data and 288 bytes of auxiliary data for additional error detection and error correction.

The Standard

In the early days, CD-ROM data was stored in a variety of formats, usually proprietary to a specific CD-ROM database vendor. Some formats were so distinctive that the disks were only compatible with certain CD-ROM drives.

The High Sierra Group (HSG) developed a standard for CD-ROM data storage, and this became established (with minor changes) as the International Standard Organisation standard 9660. CD-ROM disks produced in accordance with this standard can be accessed using normal file system facilities.

Note that the standard applies to the first track on the CD-ROM. There can also be subsequent tracks, but these would be in other formats: audio, for example.

The standard specifies the volume and file structure of CD-ROM disks, including the attributes of the volume and the descriptors recorded on it, the relationship among volumes of a volume set, the placement of files, the attributes of the files, and the record structures for applications programs. The standard also specifies the functions to be provided within systems which are intended to originate or receive CD-ROM disks. It is not intended to be specific to any one operating environment, but is envisaged as a means of information exchange between systems.

In the standard, the sectors of a volume are organised into logical sectors normally of 2048 bytes, though larger logical sectors (consisting of multiple physical sectors) are also supported. The set of logical sectors constitute the volume space. The first 16 sectors (0-15) are reserved as the system area, which is not further defined. The remaining logical sectors form the data area. The volume space is organised into logical blocks of 512 or greater, but not exceeding the logical sector size.

File sections are recorded in the data area, which also contains volume descriptors, file descriptors, directory descriptors and path tables. Each file section is recorded as an extent and identified by a descriptor in a directory. An extended attribute record can be associated with the file section. Each directory is recorded as a file in a single extent, and identified by a directory descriptor in another directory or in a volume descriptor. Each directory is also identified by a record in a path table which in turn is identified by a volume descriptor. Space within the data area may also be assigned to one or more volume partitions.

The directory records are sequenced by file name and version number. A hierarchical relationship exists between the root directory and all other directories. The descriptors utilise d-characters and a-characters for file names and textual items. D-characters are numerics (0-9), upper case alphabetics (A-Z) and the underscore character. A-characters include all the d-characters plus some additional special characters (Figure 1).

A file identifier consists of a file name, and optional separator period, file extension, semicolon and file version number

D-CHARACTERS	0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z _
A-CHARACTERS	0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z _ sp ! " & ' ( ) * + , - . / : ; < = > ?

Figure 1 - A-characters and D-characters

EUROPEAN INFORMIX USERS GROUP

Informix is a trademark of Informix Software Inc.

Forthcoming Seminars to be held in Central London

16th May 1991	***Interoperability***
12th Sept 1991	***Multimedia systems***
5th Dec 1991	***Complex databases***

For further details contact:

Bryce Campbell	***(0532) 738 244***
Jeremy Russel	***(081) 446 6481***
Jerry Nichols	***(071) 251 2128***

CIRCLE NO. 787

PRO-DIR V3.0

The Power Directory Utility for the Professional Programmer

NO MENUS !!!
45 page manual and on-line help included

PRO-DIR brings you the power of a command line directory utility with unsurpassed search and selection features. It combines some of the best features seen on DOS, UNIX, VMS etc. at an affordable price of just £49 (including VAT and P&P).

PRO-DIR makes the task of finding and managing files effortless, even on large directory structures. You can now find those long lost files on your network drive with commands like

```
d  G:\USER\BOB\...\TAX*R*T*N.*    after=01-01-87    before=-365
```

Look how easy it is to recover disk space when your hard disk is full and you need to free a few megabytes quickly - the command

```
d  C:\,D:\,E:\    +R    min=50k    since=-7
```

finds all files larger than 50kb that you created over the last seven days *anywhere* on drives C, D or E. (Also see below how the +B switch can help).

TAKE A LOOK AT THESE FEATURES

d *A*B*02 E?E	Extended wildcards in all file and directory specs
d *.EXE,*.BAT,*.COM	Multiple unlimited file *select* specs
d *.* *.EXE,*.BAT	Multiple unlimited file *exclude* specs
d C:**S*R*C**.C	Extended wildcard directory specification
d C:*.C,D:\USER*.C	Searching on multiple drives (network drives are supported) and directories
d C:\USER*.C +R	Recursively find all matching files below specified directory
d ***.C	Search for files that are a specified number of levels deep
d \A*\\...*S*R*C*\\...*.C	Search for files with an unknown number of intervening sub-directories
d +S.f	Sort by filename (or file extension, size, date, attribute)
d max=2M min=50k	Select by file size range
d on=-7	Select by relative (or absolute) date. Relative dates are from current date
d after=-365 before=-31	Select by relative date range
d after=10-12-90	Select by absolute date range
d a=h n=d	Select/exclude by file attribute (eg d=directory file, h=hidden file)
d +p	Comprehensive display paging options
d +B	Generate output in batch file format for further processing
d +T	Display the total disk usage of selected files
d +h	Display hidden files
d +m	Display the seconds field of the selected files
d +ft-asd	Enable or disable display fields (eg f=filename field, s=file size field etc.)
d +?	Display the on-line help screens
set PDOPTIONS=+pt-h	Complete customization of program default settings via the set command

You can combine all these features together. You may specify as complicated a command as you wish, provided it fits on the DOS command line! And that's not all - we haven't even listed all the switches and options that are available. Call or fax us for more information. Alternatively, send us the coupon below, or call us with your order 9AM - 5.30PM.

Please send me ____ copies of PRO-DIR at £49 per copy including VAT and P&P.
Name: ....
(Company): ....
Address: ....
....
Access/Visa number: .... Expiry date: ....
Please specify disk format: ___ 5.25" (360kb) OR ___ 3.5" (720kb)
If paying by cheque, please make it payable to Korala Software. Thank you

KORALA SOFTWARE
Korala Associates Ltd
FREEPOST
Edinburgh EH3 0EP
Phone: 031 556 9208
Fax: 031 556 9215

Suitable for IBM PC, XT, AT or 100% compatibles running DOS 3.0 or higher. Please allow 28 days for delivery. International orders: No VAT, postage charged at cost.

CIRCLE NO. 788

ADEPT - Skilful, expert and highly productive

G.U.I.s / RDBMS Interface / Expert System Shells

Whether you are searching for a ready-made solution or seeking advanced tools to build an application, an ADEPT product can make your task easier. . . .

Advanced Development Environment Programming Tools

The ADEPT family of products has been developed for Open Systems and designed to provide a highly flexible development environment that allows developers to efficiently produce high quality applications. Object-orientated and offering superb graphic capabilities, ADEPT products are fully compatible with Open Look and Motif.

Proven Technology

Developed by ILOG, France's leading supplier of AI software, ADEPT products are used by over 3000 users in 17 countries. They include GUI development tools, MASAÏ and AÏDA; ASQUELL, an intelligent RDBMS interface; and SMECI, an expert system shell. Now they are available in the UK for the first time.

ADEPT products allow developers and users to be productive, deliver results and stay competitive.

Come and see how ADEPT can revolutionise your application at
Stand 109, Software Tools '91 11-13 June
or at
Stand 212, European UNIX Show 1991 18-20 June

SIGNAL
COMPUTING LIMITED
THE FUTURE OF COMPUTING TECHNOLOGY

If you would like to learn more about ADEPT please contact us for information or to arrange a demonstration.
Signal Computing Ltd, 20 Nugent Road, The Surrey Research Park, Guildford, GU2 5AF U.K. Tel: +44 483 579900

CIRCLE NO. 789

(1-32767). The filenames can be up to 31 characters long. However, these should also obey the rules of the target system for which the disk is intended - for example an MS-DOS system should use the standard <8 chars>.<3 chars> format. File names must be made from d-characters.

Microsoft extensions

To support the standard in the MS-DOS environment, Microsoft has developed specifications for two separate pieces of software. The first is the lower-level device driver written specifically for a particular CD-ROM drive, usually as a .SYS file that is loaded at boot time via the CONFIG.SYS.

This driver sets up lower-level communications between the CD-ROM drive and the operating system, providing a link between the CD-ROM drive or drives and the other, higher-level piece of software, the MSCDEX.EXE file. This was actually developed by Microsoft, but is licensed to the drive manufacturers and supplied with the drives. The current version for most drives is V2.1, although IBM supplies V2.2 with its CD-ROM drive.

MSCDEX is run after the CD-ROM device driver is loaded. First, MSCDEX assigns specific operating-system drive identifiers to each CD-ROM drive. It then works with the device driver to redirect operating-system disk read requests to the CD-ROM drive. In addition, MSCDEX sets up a disk buffer in RAM to speed access to the relatively slow CD-ROM disk drives. The size of the buffer is specified on the MSCDEX command line. Note that the LASTDRIVE entry will be required in the CONFIG.SYS if a letter higher than E: is to be assigned to the CD-ROM drive.

Accessing CD-ROM

CD-ROM device drivers are a hybrid of block and char device drivers. MSCDEX interfaces with MS-DOS for all I/O, and does not use the BIOS. The CD-ROM is treated as if it were a network drive. This means that when the device driver and MSCDEX have been installed, you can access any CD-ROM disk that meets the standard, using normal MS-DOS commands such as DIR.

You cannot, of course, expect to use commands that write to the drive, so it is no surprise that FORMAT fails with the message 'Cannot FORMAT a Network drive'. Less obviously, CHKDSK will fail with a similar message.

It is essential to bear in mind the restrictions imposed by the standard. For example, file names must contain upper-case characters. For MS-DOS, this is no problem, since filenames are mapped to upper-case before they are looked up. However, the length of the path name could be a problem since the standard allows 255 characters while MS-DOS is limited to 64 characters. The disk is, of course, read-only. If your application needs to create temporary files, it may be necessary to specify a working directory on a fixed disk. In Computer Library, the working directory is selected as the default, and the CD-ROM application is started eg by typing G:\CL, where the drive letter indicates the CD-ROM drive.

Some CD-ROM Disks

ES/9000 PROCESSOR COMPLEX CUSTOMER LIBRARY:
This disk was published by IBM to the ISO 9660 standard. It contains softcopy versions of planning and operating guides for the IBM ES/9000. The publications can be individually accessed by file name, but a version of the IBM BookManager program is required for viewing.

As different versions of the publications become available at later system engineering change (SEC) levels, the CD-ROM will be updated. Because back-level publications will also be maintained on subsequent CD-ROMs, it will not be necessary to retain old CD-ROM disks.

COMPUTER LIBRARY:
This disk is a computer periodicals database published by the Ziff Communications Company. The disk meets the ISO 9660 standard, and contains 12 month's worth of articles from leading computer publications such as .EXE and Byte. The October '90 edition, for example, includes over 55,000 articles and abstracts from more than 140 publications.

Computer Library allows you to retrieve information using the Lotus BlueFish Searchware. You enter words, phrases, or terms you want to find and the searchware locates every document in which these definitions occur. You can read, copy, and print the documents.

CD-ROM specific requests

Not all CD-ROM requirements can be handled through standard MS-DOS commands, especially if there are additional audio tracks recorded on the CD-ROM disk. It is possible to access the CD-ROM device driver by issuing commands through the drive adapter. With the IBM SCSI CD-ROM and SCSI adapter, for example, several SCSI commands are supported (inquiry, read capacity, read extended and request sense), while a series of CD-ROM specific commands can be routed through the adapter command to send other SCSI commands. However, to avoid re-entrancy problems, and to allow MSCDEX to monitor all media changes, all applications that wish to communicate with CD-ROM device drives are recommended to use the MSCDEX interface provided for function requests.

Access to the functions is through an INT 2Fh interface. Register AH contains 15h to differentiate MSCDEX requests from those of other INT 2Fh handlers. Register AL contains the code of the function to be performed.

The sample program

The CDROM.COM program listed in Figure 2 illustrates the use of the MSCDEX interface. This program was assembled using the IBM Macro Assembler/2, LINKed and converted to .COM form using EXE2BIN.

The first MSCDEX function request in this program (beginning of START PROC) is to get the number of CD-ROM drive letters. This could be greater than one, for multiple CD-ROM drive configurations, or for CD-ROM drives that support subunits (such as juke-box style CD-ROM players). If BX remains zero, this indicates that MSCDEX is not installed. The first (or only) drive letter assigned to CD-ROM drives is indicated by CX (0=A,1=B,2=C etc). You can also issue a further request to provide the full list of drive numbers. These need not be sequential since MSCDEX allows the selection of specific drive letters for a device, or you may interleave network devices.

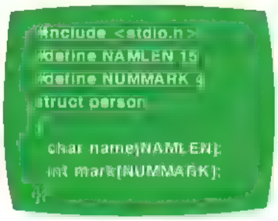
In the program, subsequent function requests use the initial drive number provided, since it is assumed that only one drive is installed.

The second function request gets the version of MSCDEX installed. For versions earlier than V2.00, this function was not supported, and so BX will remain zero. Procedure BTOASC is used to convert the binary result into displayable ASCII form.

Take Our Course In C And The First Lesson You'll Learn Is In Economics.

C's power and portability make it the language of choice for software developers.

Unfortunately, learning C can be a very costly proposition. Classroom instruction is, in a word, expensive. And many C video courses carry hefty price tags.



The top C video course at the lowest possible price

But now, there's The Complete C Video Course from Zortech.
It's the ultimate C training tool for home or work. And all it costs is £199.95.

You get ten videos with 36 lessons covering all levels of programming skill. A comprehensive, easy-to-follow 365 page workbook. And even a free C compiler.

Free C compiler included

Yes, that's right. The Complete C Video Course includes our famous C compiler (it runs on any MS-DOS machine) with linker, library manager, full graphics library and on-line help. It's the choice of professional programmers everywhere for fast code, fast development and fast debugging.

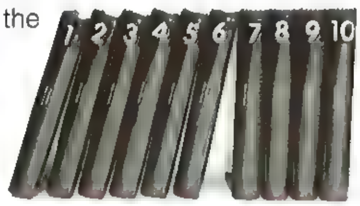
Learn C in as little as two weeks

Speaking of speedy, with The Complete C Video Course you can learn C in only two weeks.
Compare that with the up to four months it can take to learn C in class.

Each lesson averages 17 minutes of clear, concise instructions. Used in conjunction with our workbook you'll find they provide everything you need to know to become proficient in programming in C.

Save your company thousands

If you think The Complete C Video Course is a great way for you to save money learning C, think about how much it could save your company. Use it instead of sending programmers to school and you'll save thousands. What's more, The Complete C Video Course is even tax deductible.
C is unquestionably the most valuable programming language you can master. And now you can get everything you need to become productive in it from course to compiler to tools for an economical £199.95. Mail the coupon or call our hotline to receive it ASAP.



ZORTECH

Look at all these C video pluses

- Only £199.95 complete
- Ten videos with 36 lessons.
- Comprehensive 365-page workbook.
- Free C compiler with linker, library manager, full graphics library and on-line help.
- Compiler and hardware independent.
- Designed to help you learn C in as little as two weeks.
- Tax deductible.

THE BRITISH PERSONAL COMPUTER AWARDS 1988
PERSONAL COMPUTER WORLD EDITOR'S AWARD
WINNER

Zortech Limited,
106-108 Powis Street,
London SE18 6LU.
Tel: 01-316 7777
Fax: 01-316 4138

★ Yes, rush me The Complete C Video Course including free C compiler for £199.95 (VHS only)
★ Please include (No.) extra workbooks at £19.95 each.
★ I'd like to order (No.) extra C compilers with this course at a low price of £49.95

Name/Company..................

Address........................

....

....

Post Code.....................

Phone.........................

Here's my cheque for............

VISA/ACCESS No................

Exp. Date......................

All prices exclusive of VAT and post & packing.

The Complete C Video Course £199.95

Order Hotline 01-316 7777

CIRCLE NO. 790

```
        TITLE   CDROM
;
;       CDROM (Test MSCDEX function requests)
;       Michael Price    .EXE
;
DOSINT  MACRO   FUNCTION
;CALL DOS INTERRUPT 21H
        MOV     AH, FUNCTION
        INT     21H
        ENDM
;
CSEG    SEGMENT PARA PUBLIC 'CODE'
        ASSUME CS:CSEG,DS:CSEG,SS:CSEG,ES:CSEG
;
        ORG     100H
CDROM:  JMP     START
;COM FILE ENTRY ALWAYS AT 100H
;
CPYRITE DB      'CDROM Michael Price',1AH
MESG1   DB      'MSCDEX NOT INSTALLED$'
MESG2   DB      'MSCDEX VERSION IS:      '
ASC     DB      2 DUP (?),'.',2 DUP (0),
        DB      0DH,0AH,'$'
;DRIVE LETTER (0=A,1=B,2=C ETC)
DRIVE   DW      0
CPYMSG  DB      'COPYRIGHT FILE:          '
COPYRT  DB      38 DUP (0)
        DB      0DH,0AH,'$'
ABSMSG  DB      'ABSTRACT FILE:           '
ABSTRC  DB      38 DUP (0)
        DB      0DH,0AH,'$'
VTOC    DB      2048 DUP (0)
        DB      0DH,0AH,'$'
CRTMSG  DB      'CREATION DATE/TIME:      '
CREATE  DB      17 DUP (45)
        DB      0DH,0AH,'$'
MODMSG  DB      'MODIFICATION DATE/TIME: '
MODIFY  DB      17 DUP (45)
        DB      0DH,0AH,'$'
VSIMSG  DB      'VOLUME SET IDENTIFIER:  '
VSIDEN  DB      128 DUP (45)
        DB      0DH,0AH,'$'
PUBMSG  DB      'PUBLISHER IDENTIFIER:   '

PUBLIS  DB      128 DUP (45)
        DB      0DH,0AH,'$'
;
START   PROC    NEAR
;
;GET NUMBER OF CD-ROM DRIVES
        MOV     AL,00H
        MOV     AH,15H
        INT     2FH       ;MSCDEX HANDLER
        CMP     BX,0H
        JNE     INSTL     ;MSCDEX NOT INSTALLED
        MOV     DX,OFFSET MESG1
        DOSINT  09H
        MOV     AX,16     ;SET RETURN CODE
        DOSINT  4CH       ;QUIT
INSTL:  MOV     DRIVE,CX
;
;GET MSCDEX VERSION
        MOV     AL,0CH
        MOV     AH,15H

        MOV     BX,0

        INT     2FH

        MOV     AX,BX     ;BINARY TO ASCII
        CALL    BTOASC
        MOV     DX,OFFSET MESG2
        DOSINT  09H
;
;GET COPYRIGHT FILE NAME
        MOV     AL,02H
        MOV     AH,15H
        MOV     BX,OFFSET COPYRT

        MOV     CX,DRIVE
        INT     2FH
        MOV     DX,OFFSET CPYMSG
        DOSINT  09H
;
;GET ABSTRACT FILE NAME
        MOV     AL,03H
        MOV     AH,15H
        MOV     BX,OFFSET ABSTRC
        MOV     CX,DRIVE
        INT     2FH
        MOV     DX,OFFSET ABSMSG
        DOSINT  09H
;
;READ VTOC
        MOV     AL,05H
        MOV     AH,15H
        MOV     BX,OFFSET VTOC
        MOV     CX,DRIVE
        MOV     DX,0
        INT     2FH
        CLD
;GET CREATION DATE
        MOV     SI,OFFSET VTOC+813
        MOV     DI,OFFSET CREATE
        MOV     CX,17
        REP     MOVSB
        MOV     DX,OFFSET CRTMSG
        DOSINT  09H
        CLD
;GET MODIFICATION DATE
        MOV     SI,OFFSET VTOC+830
        MOV     DI,OFFSET MODIFY

        MOV     CX,17

        REP     MOVSB

        MOV     DX,OFFSET MODMSG
        DOSINT  09H
        CLD
 ;GET VOLUME SET IDENTIFIER
        MOV     SI,OFFSET VTOC+190
        MOV     DI,OFFSET VSIDEN

        MOV     CX,128

        REP     MOVSB

        MOV     DX,OFFSET VSIMSG
        DOSINT  09H
        CLD
;GET PUBLISHER IDENTIFIER
        MOV     SI,OFFSET VTOC+318
        MOV     DI,OFFSET PUBLIS
        MOV     CX,128
        REP     MOVSB
        MOV     DX,OFFSET PUBMSG
        DOSINT  09H
;
        MOV     AX,0
        DOSINT  4CH                        ;END
START   ENDP
;


BTOASC  PROC
;CONVERT BINARY AX TO ASCII
        MOV     CX,AX
        MOV     AL,CH
        CBW             ;VERSION
        SUB     DX,DX
        MOV     BL,10
        DIV     BL
        ADD     AL,30H
        ADD     AH,30H
        MOV     ASC[0],AL
        MOV     ASC[1],AH
        MOV     AL,CL
        CBW             ;SUBVERSION
        SUB     DX,DX
        MOV     BL,10
        DIV     BL
        ADD     AL,30H

        ADD     AH,30H

        MOV     ASC[3],AL

        MOV     ASC[4],AH

        RET

BTOASC  ENDP ;
;
CSEG    ENDS
        END     CDROM
```

Figure 2 - CDROM.COM listing

dBASE USERS!

SOFTWARE TOOLS

Software Tools '91

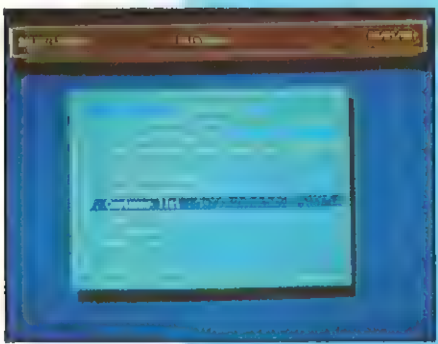
11-13 June
Wembley Conference and Exhibition Centre
London

STAND No. 35

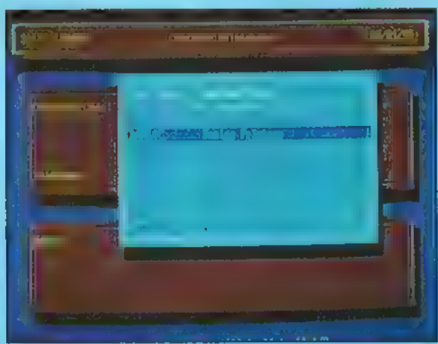
How many days would it take to program, test and debug this system? With Sycero and Clipper you could do it in HOURS!

Clipper is the applications development standard for PC database management systems. Clipper's open architecture and superior networking capabilities let you create more sophisticated and powerful applications.

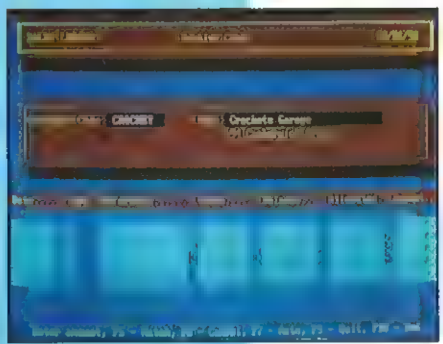
Sycero is a complete applications generator for Clipper. Making use of all of Clipper's advanced facilities, Sycero lets you develop PC applications in a fraction of the time it would take to do manually.



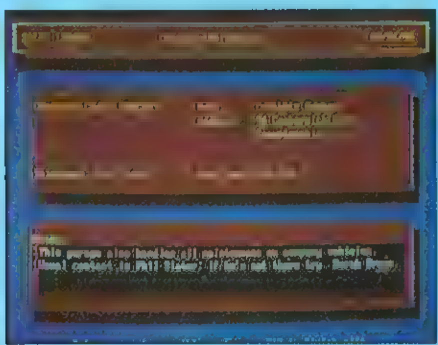
Pulldown, bouncebar and Novell style menus produced in minutes.



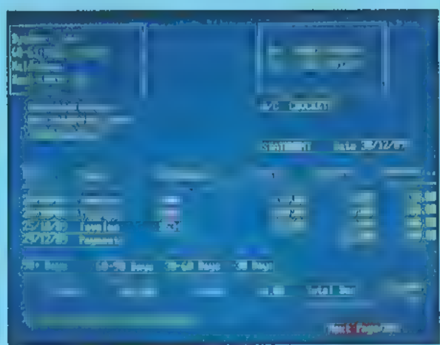
Overlay a browse window with a function key in just five lines of code.



Automated transaction handling.



Memo fields give you word processing in a window on your screens.



Complete & powerful report generator for forms, lists and more complex invoices/statements.

Sycero with Clipper – the perfect development environment.

Please send me further details about Sycero & Clipper:
Name: ....
Address: ....
....
....
Tel: .... Fax: ....

System C Ltd., 60-61 High Street, Maidstone, Kent ME14 1SR
Tel:0622 691616 EXE 5/91 **Fax:0622 691241**

Nantucket

CIRCLE NO. 795

System C

The next function request is for the copyright file name, which is then displayed on the screen. The abstract file name is obtained and listed in a similar manner. Both items are actually contained in the primary partition descriptor block. There is a specific function request to read the volume table of contents, and this brings in the whole of the descriptor block.

Numerical entries in the descriptor are in both Intel format (least significant byte first) and Motorola format. For example, the logical block size (2048 decimal) is specified in four bytes as 00080800, representing a hex value of 0800.

The program selects and displays the creation date and time, the modification date and time, the volume set identifier and the publisher identifier.

Figure 3 shows the output from the program for the IBM ES9000 Customer Library CD-ROM, along with the table of contents (TOC) for the disk. This shows that the disk was formatted to the ISO 9660 standard.

The program was also run with a variety of CD-ROM application disks, and it became clear that the ISO 9660 standard was not always fully or correctly implemented. The VTOC for the Computer Library, for example, held similar information but at different offsets. It is evidently unwise to rely on complete compatibility with the standard.

Other function requests

Among the other functions supported by the 2Fh interface is an absolute disk read which corresponds to the MS-DOS INT 25h. It will be converted into a READ_LONG device driver request. This function can be used to read non-standard disks also. A matching absolute disk write has been defined in the MSCDEX specifications for future use in CD-ROM authoring systems, but it is not implemented in current versions of MSCDEX. There is a request to get directory entries, and these can be returned in the standard form for HSG or for ISO 9660 directories. The differences lie in the provision in the ISO standard of an additional byte of date and time, used for a Greenwich mean time offset.

Finally, there is a send device driver request that is intended to simplify communication with CD-ROM device drivers. This allows a

```
Track      Start        Length      Track type
  1       00:02:00     04:22:30       data
Disc has 1 track    - Lead out begins at 04:24:30
Data tracks formatted to ISO-9660 standards.

TABLE OF CONTENTS (TOC)

MSCDEX VERSION IS:        02.02
COPYRIGHT FILE:           LICENSE.AGR
ABSTRACT FILE:            README
CREATION DATE/TIME:       1990090500000000
MODIFICATION DATE/TIME:   1990090519010000
VOLUME SET IDENTIFIER:    SK2T668500
PUBLISHER IDENTIFIER:     IBM CORPORATION

PRIMARY VOLUME DESCRIPTOR
```

Figure 3 - Output from test program

variety of facilities to control the CD-ROM drive, including start and stop audio, lock and unlock the drive door, eject or accept the current CD-ROM tray, get disk table of contents and so forth. There is also a function to write device control strings, thus allowing the application to access device specific features that may not be directly supported.

Optimising performance

The MSCDEX parameters allow you to specify a cache for sectors. If this is made large enough to hold the number of sectors normally required, then performance will be improved. However, it is necessary to consider the effect of directory size. Directories are always a multiple of the logical sector size (usually 2 KB), so to open a file located in a large directory could involve scanning many directory sectors before the file is located, thus flushing the cache. Ideally, directories should have no more than 40 entries, to keep their size down to a single sector. Further performance benefits will result if related files are grouped together, since this maximises the chance of the required subdirectory already being in the cache.

Of course, unless you are producing your own CD-ROM disk, you will not have control over the placement of files. An understanding of the structure of the CD-ROM data may allow you to bypass the relatively slow MS-DOS file searches and maintain your own index of data sectors, reading the data directly by absolute sector reads.

Perhaps the best strategy, however, is to use the fixed disk as a form of secondary cache, and store the most used files in temporary space. This approach will allow you to use the standard file system for your chosen environment and make it easier to port your application to new environments.

Conclusion

CD-ROM disks are now being produced to ISO 9660 standards. You can use the Microsoft CD ROM Extensions to access the disk from your application, using normal file handling facilities, or take advantage of the underlying data structure to provide higher performance. In either case, since the CD-ROM disk will be the same for other operating environments, portability of applications using CD-ROM is increased.

In this article, I have concentrated on the MS-DOS environment. In a future article, I hope to look at the way the same facilities are implemented in the OS/2 environment, using the compact disk installable file system, and also seeing how Microsoft and others are developing the CD-ROM concept through CD-ROM XA (Extended Architecture) to support digital audio and still images.

EXE

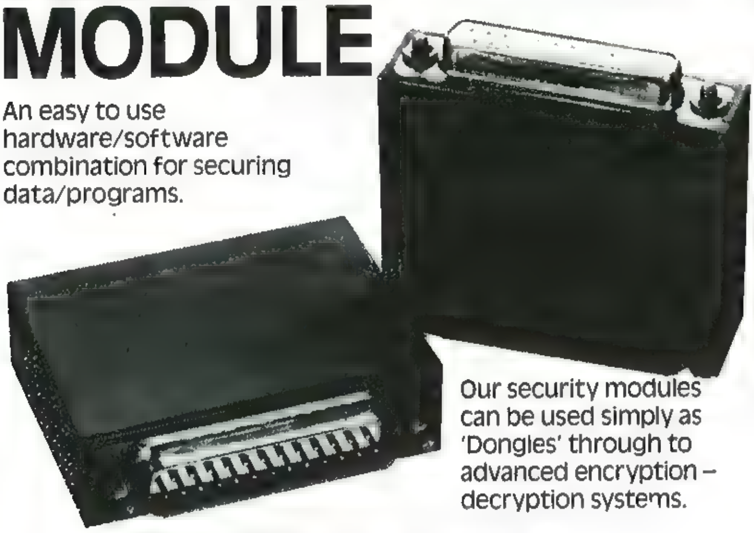
Michael Price is a systems design consultant in the financial industry arena. He takes a particular interest in CD-ROM and is developing in his spare moments a utilities package for CD-ROM.

The Computer Library was provided for review by Optech Limited (0252-714340). It is available on annual subscription with bi-monthly updates for £895.

Further information on CD-ROM technology and the ISO 9660 standard can be found in 'CD-ROMS: Breakthrough in Information Storage' by Frederick Holtz, published by Tab Books.

SOFTWARE SECURITY MODULE

An easy to use hardware/software combination for securing data/programs.



Our security modules can be used simply as 'Dongles' through to advanced encryption – decryption systems.

- ★ For IBM PC family (and compatibles)
- ★ Variants for other systems
- ★ Advanced construction – rugged and compact
- ★ Proven throughout the world
- ★ Transparent operation – user friendly
- ★ Driver software supplied for variety of applications

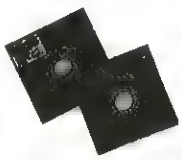
CTL

Control Telemetry of London
11 Canfield Place, London NW6 3BT
Tel: 071-328 1155 Fax: 071-328 9149

CIRCLE NO. 799

BRENT
Communication and Telemetry Consultants

"MaxPro – the ultimate in hardware for software protection"



The MaxPro Data key fitted to the Parallel Port is totally transparent to the end user.

Software customers make as many backups as are required but of course only the machine using the individually coded MaxPro key will run the program.

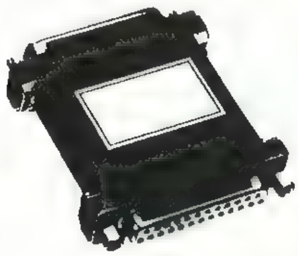
BEWARE THE PIRATE'S PATCH
High security encryption is created by the Simple-to-Use Menu-driven MaxPro attachment software. Other data keys may be patched out in the software rending it totally vulnerable. MAXPRO protected files cannot be patched. Choose whether or not to shut down the program if tampering occurs.

Optional limited life and Usage information collection facilities are provided.

No source or object code modules needed.

The MaxPro system works on IBM; PC; XT; AT Compatibles and PS2.

For additional information contact us at:



Brent Communications
Unit 2
Dragon Industrial Estate
Harrogate HG1 5DN

Tel: (0423) 566972
Fax: (0423) 501442

CIRCLE NO. 800

System Science

C COMPILERS	
Microsoft C 6.0 with PWB	£245.00
Quick C & Quick Asm 2.51	£99.00
Quick C ver 2.5	£65.00
Borland C++	£245.00
Zortech C++ Dev Ed	£275.00
Watcom C 8.0	£255.00
Aztec C86 Comm	£265.00
C compilers for 386	£call

C GRAPHICS & DATAFILE	
Halo	£195.00
GSS GDT	£445 00
Metawindows	£175.00
Essential Graphics (no roy)	£275.00
PCX Prog Toolkit (new 4.0)	£165.00

C DATA & MISC	
Btrieve for DOS	£345.00
XQL for Btrieve	£475.00
Ctree (source)	£195.00
CodeBase 4.2 (source)	£175.00
db__VISTA III	£545.00
GL Financial Mathlib	£245.00
Blaise C Tools Plus	£89.00

C SCREENS & COMMS	
Cscape, Look & Feel	£365.00
Vermont Views & Designer	£325.00
Panel Plus II	£275.00
C Worthy & cwARCHITECT	£call
Essential Comms (src)	£195.00
Blaise C Aynch Manager	£115.00
Greenleaf CommLib	£185.00

PROGRAMMERS TOOLS	
SPE prog editor	£189.00
MKS Toolkit (Unix toolset)	£155.00
PVCS Prof (lower price)	£375.00
PolyMake	£135.00
vsDesigner (Case Tool)	£475.00
Brief Editor 3.0	£195.00
Kedit Editor	£110.00
Flow Charting 3	£150.00
Dan Bricklin DEMO II (new)	£195.00
MS Progs Lib 1.2 CD	£225.00
Personal Rexx	£110.00

ASSEMBLERS & LINKERS	
MS Macro ASM 5.1	£90.00
Turbo Debugger & ASM 2.0	£99.00
RTLink Plus	£345.00
Plink 86 Plus	£295.00
Sourcer & Bios (DisAsm)	£125.00

CROSS DEVELOPMENT	
Introl C or Modula-2	£1445.00
Aztec C Cross	£275.00
IAR C Cross with CSpy	£call
2500AD Cross AsM	£135.00
Simulators & others	£call

FORTRAN	
MS Fortran-77 5.0	£235.00
Lahey F77L-EM/32 & 386 OS	£845.00
Salford FTN77/386	£765.00
Lahey F77L	£375.00
RM Fortran Forte	£475 00

PASCAL, BASIC	
Turbo Pascal (6.0)	£75.00
Object Professional (for TP)	£110.00
MS Quick Pascal	£60.00
Blaise Power Tools Plus	£89.00
Blaise Asynch Plus	£89.00
MS Basic 7.1 with PWB	£255.00
MS Quick Basic 4.5	£65.00
PowerBasic (ex Turbo Basic)	£85.00
many Pascal, Basic libraries & tools	£call

OTHER LANGUAGES	
Smalltalk V/Windows	£295.00
TopSpeed Modula-2	£145.00
MS Cobol 4.0 (new ver)	£445.00
RM Cobol-85 ver 5.0	£995.00
LMI ur Fresh Forth	£225.00
PC Logo	£50.00
Clipper 5.0	£395.00

MEMORY, MATHS etc	
386 Max 5.0	£75.00
QEMM/386 5.1	£60.00
QRAM for 286	£55.00
Above Disc (LIM Emul)	£69.00
Norton Utilities (new ver 5)	£110.00
Spin Rite II	£69.00
Derive (Symbol Alg) 2.0	£135.00
MicroStat II	£295.00
Mathematica by Wolfram	£call

WINDOWS 3.0	
MS Windows 3.0	£85.00
MS Windows SDK & MS-C	£425.00
MS Windows SDK 3.0	£325.00
Case: W	£CALL
WinPro/3 from Xian	£525.00
WindowsMaker	£595.00
Toolbook by Asymetrix	£295.00

"OUR RANGE INCLUDES MANY MORE ITEMS FOR DOS, OS/2, UNIX ETC. PLEASE ENQUIRE"

3-5 Cynthia Street,
London N1 9LF
Fax: 071-837 6411

BARCLAYCARD VISA

Access

Prices are subject to change and alteration Prices include UK Delivery but are exclusive of VAT

071-833 1022

CIRCLE NO. 801

He called for his pipe

Under MS-DOS, it is good for MORE, and SORT, and pretty well nothing else. Under UNIX, you can write whole programs to exploit the pipe, as Peter Collinson explains.

If you are an MS-DOS or UNIX user, then you will be familiar with the idea of a pair of commands grouped by using a vertical bar:

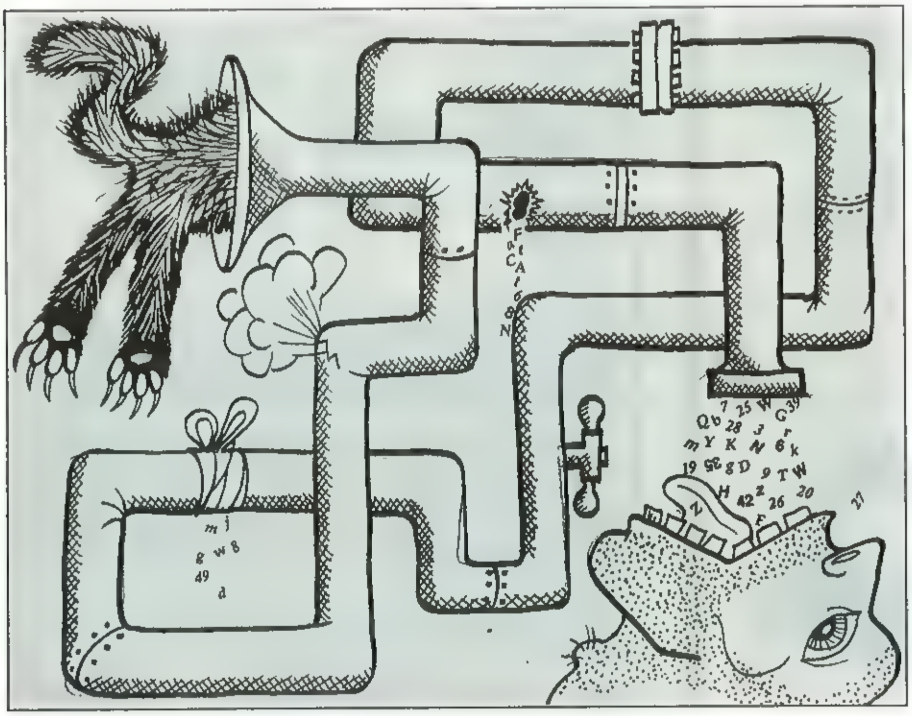
```
command1 | command2
```

This means that the first command is run with its output being inserted as the input to the second command. MS-DOS stole the idea from UNIX and implemented it by making the first command create a temporary file. When the first command dies, the second command is started with its input being taken from the temporary file. The vertical bar is a convenience for the user, who no longer needs to worry about intermediate results.

On UNIX, the two commands are run in parallel. Of course, this is an illusion provided by the operating system. We should think that the commands are run in parallel, even if we know that this cannot be the case on a single CPU. The commands are started at the same time and run together. The connection between the processes is a *pipe*, an anonymous byte stream.

The pipe has two ends, a reader and a writer. The shell arranges that the standard output of `command1` is connected to the write end of a pipe, and the standard input of `command2` will be taking data from the read end of the pipe. So, all output from the the first command is sent into the pipe. It is buffered there until `command2` decides to read it. If the pipe gets too full, `command1` will be blocked from writing. *Blocked* here means that the operating system will stop it from returning from the write system call until the data transfer is completed.

Notice that in most circumstances, `command1` will not know that it has been blocked unless it takes special action. Also, it has no idea where it is writing. It is simply throwing bytes down the standard output. In the same way, `command2` has no knowledge that it is reading from a pipe. If there is no data, then the process will be blocked in the `read` system call waiting for it to appear. The system call will return to the calling process when data becomes available.



It is important for the pipe to have some buffering. It is this buffering that makes it easy for the kernel to provide inter-process synchronisation. The actual size of the buffer is system dependent - my Sun allows 4096 bytes. The buffering permits considerable parallelism in the operation of the data producer and consumer.

In general, the producer will do a `write` system call to send the data to the consumer. If the pipe can take all the data, the producer can be released to generate some more output while the consumer is reading the buffered information in the pipe. The normal steady state of the consumer is to wait for data to appear in the pipe. Once the data appears, the consumer can start working on it. Both processes can be running together, and this can speed data throughput.

The fact that this is all completely free is one of the nice things about it. Neither of the processes need to be aware that they are dealing with pipes. The programs are just following normal actions of producers and consumers The pipe buffering just means that they interact well together without any special coding being needed.

Simple programming

How easy is it to use this? Can I easily make my program automatically call a pager, like `more` or `pg` and then pass all my output through it? If you are using the standard I/O library, the routine `popen` can handle this, just like `system` can deal with calls to other programs.

To call `more`, insert a routine sequence at the start of the program like:

```
fclose(stdout);
stdout = popen("more", "w");
```

There are two arguments. The first gives the command to be executed. I suppose I should have said the 'command line' to be executed, since it is passed into an invoca-

OU WON'T FIND A FTWARE ANYWHERE*

for DOS, OS/2, Mac & UNIX

AL • EXPERT SYSTEMS & A.I • GRAPHICS • HARD DISK BACKUP • INTEGRATED SOFTWARE • LANGUAGES • LINKERS • MAC SOFTWARE • MATHS & STATS • MEMORY MANAGEMENT • MISCELLANEOUS • NETWORK SOFTWARE • OCR & SCANNING • OPERATING SYSTEMS • PIM'S & HYPERTEXT • PRINT SOFTWARE

129.00	Norton Backup	69.00	Pizazz Plus	79.00	RTLINK Plus	329.00	Toolbook/Windows	257.00
125.00	Norton Commander	79.00	PKZIP/PKUNZIP	54.00	Saber Menu/DOS	89.00	Top Priority	79.00
39.00	Norton Editor	49.00	PlanPerfect	235.00	Saber Menu/Windows	449.00	Touch & Go	20.00
229.00	Norton Guides C, OS/2 etc	57.00	Playroom	37.00	SCO XENIX/386 Complete	1275.00	Tree Diagrammer	65.00
345.00	Norton Utilities v6	109.00	Plink-86 Plus	349.00	Security Guardian	229.00	Turbo Asynch Plus	139.00
75.00	NoSquint	45.00	PolyBoost II	65.00	Seemore 1-2-3	65.00	Turbo C++	125.00
595.00	Nutshell Plus	189.00	PolyMake	110.00	Service Diagnostics Kit	395.00	Turbo C++ Professional	205.00
35.00	ObjectVision	299.00	PolyShell	79.00	Show Partner F/X	245.00	Turbo Pascal	79.00
55.00	Object Professional	99.00	Popdrop Plus	75.00	ShowText	149.00	Turbo Pascal Professional	149.00
45.00	OmniPage-386/Windows	529.00	Portex Professional	117.00	Sidekick Plus	97.00	Turbo Professional	79.00
45.00	Open Access III	479.00	Power BASIC	79.00	Sidetalk	79.00	TurboCAD	129.00
35.00	Opt-Tech SORT	125.00	PowerMenu	69.00	Sideways	39.00	Twist & Shout	49.00
165.00	Option Board Deluxe	105.00	Powerpak/Windows	69.00	Sigma Plot	445.00	Type Manager/Windows	79.00
499.00	Optune	59.00	PreScript Deluxe	295.00	Silvercomm Library	189.00	Typing Tutor IV Plus	39.00
120.00	ORCAD/PCB	1395.00	Presentation Task Force	249.00	Sim City	26.08	Typografica Prime	49.00
169.00	ORCAD/SDT	495.00	Primetime Personal	75.00	SitBack	62.00	UI Programmer v2	279.00
169.00	Org Plus Advanced	85.00	PrintAPlot	165.00	SiteLock w/Anti-Virus	329.00	UI Touch & Go	235.00
299.00	Overlay() for Clipper	99.00	Printcache	95.00	Smalltalk V/286	129.00	UltraScript	125.00
67.00	P.D.Queue for 1-2-3	59.00	Printer Assist	149.00	Smalltalk V/PM	295.00	UltraScript PC Plus	245.00
239.00	PageGarden	63.00	Printmaster Plus	49.00	Smarterm 240	179.00	Ultravision	79.00
229.00	Pagemaker/Mac	569.00	PrintQ	89.00	Smart System	469.00	Vcache	53.00
469.00	Pagemaker/Windows	453.00	Printrite	85.00	Smartware II	469.00	Ventura GOLD/Windows	549.00
259.00	Paradox v3.5	379.00	Procomm Plus	72.00	Soft Kicker Plus	95.00	Vermont Views	349.00
39.00	Paradox Engine	269.00	Procomm Plus w/Viewdata	107.00	Soft Type/Windows	129.00	Virex-PC or Mac	79.00
19.00	Paradox SQL Link	369.00	Professional File	169.00	Softbytes	59.00	Virus Hunter	99.00
229.00	Pathminder Plus	79.00	Professional Write	229.00	Software Bridge	85.00	ViruSafe	79.00
85.00	PC Anywhere III/IV	110.00	Professor DOS	57.00	Software Carousel	57.00	VM/386	145.00
92.00	PC Guide/Windows	429.00	Project Calc/1-2-3	139.00	Sourcer/486 w/BIOS	125.00	VolksWriter 4	125.00
109.00	PC Paintbrush IV Plus	159.00	Project Scheduler/Windows	449.00	SourcePrint	69.00	Vopt	45.00
159.00	PC Tools Deluxe	79.00	Project Workbench	795.00	SPF/PC	169.00	Watchdog	185.00
295.00	PC Write	85.00	Prokey Plus	79.00	Spinrite II	89.00	Where in Europe	39.00
349.00	PC-Browse	55.00	PVCS Professional	449.00	Sprint	135.00	Who-What-When	159.00
59.00	PC-File 5	79.00	PVCS/LAN	1695.00	SPSS/PC+	99.00	Windows Workstation	595.00
57.00	PC-Fullbak Plus	69.00	Q&A	239.00	StatGraphics	429.00	Wingz/Windows	329.00
69.00	PC-Globe	69.00	QDOS-II	49.00	Sterling Accountant Plus	329.00	Wonder Plus	59.00
275.00	PC-Guard/D Plus	137.00	QEdit TSR	79.00	Sterling Financial Controller	429.00	Word for Word Pro	99.00
75.00	PC-Kwik Power Disk	59.00	QEMM/386 w/Manifest	67.00	Stonybrook Modula-2	229.00	WordPerfect	269.00
253.00	PC-Kwik Power Pack	83.00	Qram w/Manifest	57.00	Storyboard Plus	229.00	WordPerfect Addtn'l Users	149.00
299.00	PC-Lint	89.00	Quattro Pro	269.00	Super PC-Kwik	53.00	WordPerfect Office	339.00
99.00	PDC PROLOG for DOS	199.00	Quick Schedule Plus	59.00	Superbase 4/Windows	449.00	WordPerfect/UNIX	329.00
79.00	Pegasus S/U Modules	205.00	QuickPak Professional	129.00	Supercalc 5	75.00	WordScan Plus/Windows	795.00
89.00	Pegasus Senior Modules	429.00	R&R Report Writer	135.00	SuperFonts	125.00	Wordstar 2000+	249.00
139.00	PerFORM Pro/Windows	269.00	R&R Clipper/Fox Module	49.00	Superkey	55.00	Wordstar Pro	235.00
325.00	PerFORM/GEM	159.00	R&R Code Generator	99.00	SuperPrint/Windows	149.00	XENON II MagaBlast	21.70
299.00	Personal COBOL	129.00	R:Base for DOS	379.00	SuperProject Expert	459.00	XtreeNET/ELS	179.00
289.00	Personal Measure	69.00	Rapidfile	195.00	Surfer	339.00	XtreeNET/Advanced	299.00
329.00	Personal REXX	129.00	Reader Rabbit	35.00	Switch-It	72.00	Xtree Pro	63.00
149.00	Perspective Junior	129.00	ReadRight/Windows	429.00	Systat w/Graphics	599.00	Xtree Pro Gold	83.00
79.00	Pertmaster Advance	1639.00	Reflection 2+	199.00	System Sleuth	85.00	XYwrite III Plus	269.00
185.00	PFS:First Choice	123.00	Reflection 4+	229.00	System WinSleuth/Windows	85.00	XYWrite III Plus/LAN	445.00
139.00	PFS:First Graphics	117.00	Reflex v2	179.00	Take II Manager	79.00	ZINC Library	179.00
45.00	PFS:First Publisher	125.00	Relay Gold	179.00	Talking Windows	235.00	Zortech C++ Video Course	269.00
59.00	Picture Publisher	449.00	Remote 2 Complete	129.00	Timeline v4	479.00	Zortech C++ Developer	269.00
129.00	Pixie/Windows	169.00	RM/FORTRAN	399.00	Timeworks	85.00		

CALL US FOR ANY ITEMS NOT LISTED

LY ADVERTISED PRICE+

OUR POLICY

- All prices and manufacturer's specifications are subject to change without notice.
- All prices exclude carriage and VAT.
- All orders subject to availability.
- Please specify media size when ordering.
- Goods are not offered on a trial basis.
- Goods are offered for sale subject to our standard conditions of sale which are available on request. E&OE.

VISA

* Provide written proof of any U.K. mail-order software company who supplies a wider range of IBM PC software than we offer in our catalogue within 14 days of this advertisement and get yourself a FREE copy of PC Tools Deluxe!

+ We are always delighted to be given the opportunity to beat other nationally advertised prices (at the time of placing an order) subject to the same terms and conditions of the company being beaten.

All Trademarks and Registered Trademarks are hereby acknowledged.

CIRCLE NO. 497

SOFTWARE PARADISE

The Software Experts

Software Paradise, Avenue House, King Edward Avenue, Caerphilly, Mid Glamorgan, CF8 1HE U.K.

FREEPOST — TODAY —

NO STAMP NEEDED

(U.K. ONLY)

☐ Please send me my *FREE* 32 page copy of ***'The Discerning Person's Guide to Software'***

NAME

COMPANY

ADDRESS

POSTCODE TEL

Post now to: Software Paradise, FREEPOST, Avenue House King Edward Avenue, Caerphilly, CF8 1ZZ

EXE 5/91

CIRCLE NO. 803

tion of the shell for processing. The second argument gives the type of action that will be made on the pipe. Here we want to write to it and say `"w"`. The other option is `"r"`, for reading. The normal standard I/O library routines can be used to send the data down the pipe.

Even though the code is dealing with a normal `FILE` data structure, `pclose` must be called to finally close down the pipe rather than the normal `fclose` routine. So you may need to remember that you have used a pipe for that particular `FILE` pointer. A production program should worry a little about what happens if the `popen` fails, returning `NULL`.

Using `popen` can be effective in simple situations. But I think that it can be something of a system load: the additional call to execute the shell feels like a large burden. The routine does have some good reasons for calling the shell. First, it does this because it wants to use the user's search path to find the command that is to be executed. Second, it wants to allow the programmer to specify some shell expansion characters, should that be needed. Third, it allows a complicated command sequence to be used as a data sink or producer.

Popen problems

I am always worried that this routine calls the shell. First, it may not be needed. If the command to be used is stored in a known place, the searching for the command in the user's path is not needed. In fact, if you want to use a known command, then you should probably specify it as a full pathname to speed up access and stop much directory searching. On the other hand, it's arguable that if a user has gone to the trouble of installing a private copy of `more`, then your program should use it. You have to balance up the arguments.

Your unsuspecting program finds the Trojan horse and, Whammo! you have a root privileged shell running on the machine

If your program is running in some privileged mode, perhaps as root, then you should never allow a user to replace the command that you intend to run by one of his own. This simply creates a gaping security hole. All he does is insert a copy of a shell in his own directory, but called by the name of the command you think that you are invoking. Your unsuspecting program finds this Trojan horse and creates a root privileged process to run it. Whammo! you have a root privileged shell running on the machine.

If the target program will run as root, you should be very circumspect about exactly what commands are placed in `popen` (or `system` for that matter). Calling `more` via `popen` in a program running as root will create another great hole. The child process that is `more` will be root too, since it inherited ownership from its parent. The system buster has merely to type '!' to be handed a shell running with super-user privilege.

This should not stop you using `popen` should it be appropriate. Just be careful about which commands you think that you are running and where you are in the file system.

Doing it by steam

If you know the command that you are using and don't want to run a shell, then it is not that difficult to handcraft some code that does the job. The code to run `more` will look something like the code in Figure 1. I have added some numbered comments to give me reference points for discussion.

The first of the two routines starts a new process, running the `more` command in a child process. First, at Comment 1, it creates a pipe using the `pipe` system call. This returns two file descriptors placed in the small vector `pi`. The file descriptor in `pi[0]` is used to read from the pipe, writing is done to `pi[1]`. The pipe isn't very useful at this point.

```
FILE *
runmore()
{
    int  pi[2];
    int  pid;
    FILE *ret;

    /* Comment 1 - Pipe creation */
    if (pipe(pi) < 0)
        fatal("rarely fails\n");

    /* Comment 2 - fork */
    if ((pid = fork()) < 0)
        fatal("Cannot fork\n");

    /* we are now running two */
    /* processes in parallel */

    if (pid == 0) {

        /* Comment 3 - child code */
        /* set up standard io */

        /* set 0 to pipe read */
        dup2(pi[0], 0);
        /* close unused pipe fds */
        close(pi[0]);
        close(pi[1]);

        /*
         * 0 -> read from pipe
         * 1 -> as before
         * 2 -> as before
         */
        if (execl("/usr/bin/more",
              "more", 0) < 0)
            fatal("Can't exec\n");
        exit(-1);

        /* not reached */
        /* end of child */
    }

    /* Comment 4 - Parent code */
    /* we will write to pi[1] */
    close(pi[0]);
    ret = fdopen(pi[1], "w");
    return (ret);
}

onexit(fi)
    FILE  *fi;
{
    int  stat;

    fclose(fi);
    /* Comment 5 - Waiting for
                child to die */
    while (wait(&stat) >= 0);

}
```

Figure 1 - Simple code to run the 'more' program

At Comment 2, we fork. If things succeed, then there are two processes in running in parallel. Both ends have the file descriptors in the pipe vector, the pipe endpoints exist in both of them. The idea is that a data producer will output to the write side of the pipe while the data sink reads on the other. They will close the respective unused ends. The pipe cannot be bidirectional, it must be used in one direction or the other. What we have done is set up the communication channel between the processes, now we must condition how it is to be used.

In the child code (starting at Comment 3), we want to arrange that the call to `more` has the read side of the pipe as channel 0. This is standard input. The `dup2` call will duplicate the reader side of the pipe, the new file descriptor will be forced to be channel 0 and existing standard output closed. We then close the unused file descriptors. This is not just the tidy mind at work, we need to close the write side of the pipe so that it is *only* open in the parent. When the parent closes it, it will be its last reference. This will deliver an end of file to the input side, the child will see that and close down cleanly.

The child next execs to the `more` program. I always add a bit of fail-safe code to complain when the program cannot be found. All being well, the child now starts running `more`. After a bit the program will 'block', waiting for some data to display.

This should not stop you using popen - just be careful about which commands you think that you are running

A little bit earlier on in the story, the parent started to run in parallel with the child. The code for the parent starts at Comment 4. First, we close the unused pipe read half (line 35) and then translate the file descriptor into an open `FILE` that is passed back to the calling code. The `FILE` is used as the argument in all output statements, output from the program travels down the pipe.

When all the work is done, the parent calls the `onexit` routine, supplying it with the `FILE` pointer that was returned by `runmore`. The `onexit` code closes the descriptor. This is the last reference to the pipe, so `more` will see an end of file. The child will die - when the user has finished looking at the file. We wait for that to happen at Comment 5.

In comparison to `popen`, this code is vastly more complex. It does, however, allow much greater control of what is happening while avoiding many of the problems caused by using a shell internally in a routine.

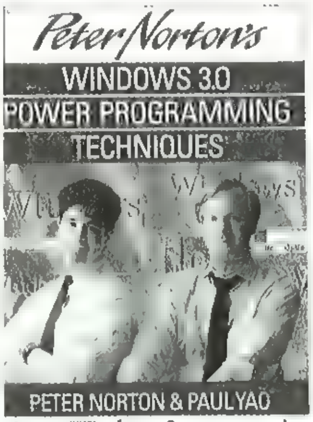
EXE

Peter Collinson is a freelance consultant specialising in UNIX. He can be reached as `pc@hillside.co.uk` *electronically (although your mailer might be happier to put the address the other way round) or by phone on 0227 761824.*

Books

The Peter Norton bandwagon turns down Windows 3 lane, and an unlikely combination of video effects and Turbo C.

Last in its line?



Peter Norton's Windows 3.0 Power Programming Techniques continues the trend of 'cult of personality' books which seem to dominate this market at the moment. Actually it's written by the eponymous PN and one Paul Yao, but the latter was presumably not considered dainty enough in shirt sleeves to have his name in the title. Complaint #2: I don't know what 'power programming' is. (What is its opposite? 'Hand programming'?)

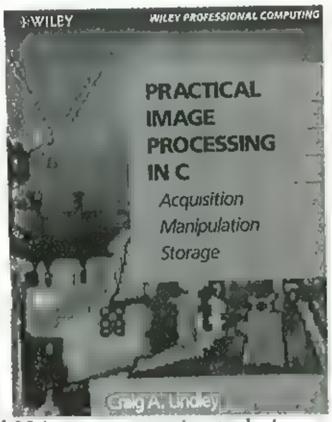
In fact, this is a rather good introduction to Windows 3 programming. The text assumes that you are equipped with a suitable C compiler (it names Microsoft, TopSpeed and Zortech), the Microsoft SDK and C programming competence. The book begins with a general background and history chapter, which includes example screen dumps from Windows versions 1 and 2 - reminding us of what we are not missing. Most of the remaining 20 chapters in this big, fat book have the same format: an example program listing or two (complete with .C, .DEF, .MAK and .RC files), plentiful mono screen dumps to show you how the thing should look, a selection of hints and tips (which suggest a depth of knowledge and wideness of experience on the part of the authors; for example, there are comparisons with Mac and a warning concerning the obsolete Windows programming practice of casting), all laid over a lively and readable tutorial narrative.

The chapters are too numerous to list, but they are organised into the following sections, which should give you some idea of the book: *An Introduction to Windows, A Minimum Windows Program, Introduction to the Graphics Device Interface, User Interface Objects, Message Driven Input, Operating System Considerations* and *More Topics in GDI Programming* (includes a chapter on printing). There are six appendices, including a list of message types and the default window procedure.

The only real problem with this book is that it is ageing fast. Perhaps it is one of the last to be published with the assumption that the only route to Windows programming is via C. With the recent launch of Turbo Pascal for Windows, an abundance of C++ class libraries, and with Microsoft threatening to launch some new Windows tools within the next few months. But, however good and high-level a Windows programming system is, it must always be an advantage to have a clear idea of what goes on at API level. This book offers a comparatively painless way of getting that knowledge.

Title: Peter Norton's Windows 3.0 Power Programming Techniques
Authors: Peter Norton and Paul Yao
Publisher: Bantam Computer Books
Pages: 939
Price: £26.99
ISBN: 0-553-34940-6

Video dreams



Practical Image Processing in C, the title read. '*The* comprehensive guide to desktop imaging', the cover added. So what's it to be, I thought: Impractical Algorithms Put Into C By Placing Curly Brackets Around Old Pseudo-Code? Or the old Comprehensive Guide ruse, aka I get to play with 50 unaffordable digitisers, then paraphrase their manuals for cash?

Actually, this book is all it claims, and more. It includes descriptions with full Turbo C source for around 20 image processing techniques. It describes a PC digitiser that you can build for £30. Throw in a full description of the TIFF format, plus explanations (with code) of Lempel-Zev & Welch (LZW)compression, and you'll begin to realise how good this book is.

It begins with a lucid description of the PC and its graphics modes. A great deal, perhaps, is skimped; but then the author, Craig A. Lindley, has greater things afoot. Still, he manages to include here an outline (complete with C driver) of the 256 colour VGA graphics modes, gamma-correction of TV colour, and, as an aside, 80x86 segment addressing and how to input through the printer port. Mr Lindley's manner is clear and concise, and he's never afraid to illustrate with code.

No time to discuss stylistic points here, though: Chapter 2 is already explaining video signals. Being an American book, NTSC (the American TV standard - Never Twice The Same Colour to us Euro-weenies) gets the analysis. The digitiser, too, is designed for the NTSC market. But, as Lindley points out, since most of the hard work is done in the (documented) PC assembler software, the modifications necessary are not too great.

So, we've built our digitiser, typed in the three resolution (320 x 200, 640 x 200, 640 x 480; all with 64 levels of grey) digitising software, added the colour software (consisting of a clever algorithm and three pieces of plastic) : what next? Well, Lindley now describes the PCX and TIFF formats, knocks out a quick library implementation for both - sadly, no room for the TIFF code, which is available only on the companion disks - and, naturally enough, moves on to printer hardcopy techniques. One colour screen dump utility with Bayer dithering later, we're at page 300, and ready to start image processing.

This consists of another 250 pages of in-depth coverage, describing techniques that begin with simple image thresholding and pseudocolouring, and finish with edge detection, filtering, and rotation. You won't be left with a Quantel Paintbox at the end of this book, but you will be left with the hardware to do most of those late '70s video effects. And the feeling you've got through a genuinely fascinating book.

Title: Practical Image Processing in C
Author: Craig A Lindley
Publisher: John Wiley
Pages: 553
Price: £32.15
ISBN: 0-471-53062-X

Books Received This Month

PC Magazine DOS Powertools by Paul Somerson	Bantam Books	£46.99	ISBN: 0-553-34938-4	pp1272
OSF/Motif - concepts and programming by Thomas Berlay	Addison-Wesley	£18.95	ISBN:0-201-54410-3	pp 490
Information Systems Methodologies - A framework for understanding by T William Olle *et al*	Addison-Wesley	£18.95	ISBN: 0-201-54443-1	pp 400
TCP/IP & NFS - Internetworking in a UNIX Environment by Michael Santifaller	Addison-Wesley	£23.95	ISBN: 0-201-54432-6	pp 235

PRE SALES SUPPORT
LONDON & BIRMINGHAM
£27K PLUS CAR

Mini/UNIX, Office Automation, Networking. Corporate Clients. No finer time to join.

SOFTWARE ENGINEERS
WORCS, COVENTRY & NORTH BIRMINGHAM
£13K TO £18K

Real-Time Industrial/Automation. C/UNIX, Graphics, Windows.

ORACLE/ACCELL/UNIFY
INGRES/INFORMIX/VAX/VNS?
WEST MIDS, WARWKS, WORCS, STAFFS.
£12K TO £27K PLUS CAR.

Vacancies from Programmer through to Project Manager. Please Post/Fax your C.V.

COBOL OR C WITH UNIX
CHESHIRE
£13K TO £25K PLUS CAR

Positions available at all levels in Development, Support and Q.A. X/Open, Windowing, Commercial Systems.

SOFTWARE SYSTEMS ENGINEERS
HEMEL HEMPSTEAD
TO £22K

P.C. UNIX/C. Device Drivers. Netware. Structured career path.

IBM/AIX EXPERIENCE?

If you are seeking a position in this area or whatever level in the West Midlands contact me now. My internationally based client needs you.

SOFTWARE/FIRMWARE
ENGINEERS
BRACKNELL
TO £24K

P.C. UNIX/C Real-Time, Automation, LANS Graphics, Networks, Device Drivers. Commitment to quality is this company's hallmark.

SOFTWARE ENGINEERS & SNPS
BRISTOL
ALL LEVELS TO £18K

Real-Time Multitasking C on PCs for technical processes.

ABACUS
CAREERS
LTD

3rd Floor, Smithfield House, Digbeth, Birmingham B5 6BS. Tel: 021-666 7484. Fax: 021-666 7430

If your skills are in the frame we can help you find a position.

Opportunities North of Watford
contact
Actis Recruitment
Tel: 0204 20200
17 Chorley New Road
BOLTON
BL1 4QR.

*Unix is a trademark of AT&T

ACTIS
RECRUITMENT

.EXE Magazine

the famous journal for software developers, is expanding its editorial staff. We are seeking a

STAFF WRITER

and a

TRAINEE STAFF WRITER

to join the small team of young people who put the magazine together each month.

The post of Staff Writer requires complete familiarity with the C programming language and either the MS-DOS or UNIX operating system, the ability to write articulately and clearly. Experience of Windows, X, LANs, OOP, dBASE and other programming skills is desirable. Points will also be awarded for enthusiasm and wit. The job includes conceiving, researching and writing feature articles, testing software for product reviews, attending press conferences and helping to plan the magazine.

The Trainee Staff Writer will have knowledge of at least one programming language. However, for this post we require a high standard of English and good proof-reading capabilities. You will be expected to write news stories, proof-read articles and generally help out the editorial staff.

The successful candidates will receive on-the-job training in the art of assembling a monthly magazine. Both jobs are fun, but demand hard work and commitment. Perks include being the first to play with the latest software tools, hob-nobbing with the Great and the Good of Software Development and having PR people suck up to you. The successful candidates will be encouraged to develop their programming and writing skills with a view to promotion to senior positions on the magazine.

The salaries offered will be £11k-£13k for the senior position, £9k-£11k for the junior, depending on experience. Please apply, with CV and a relevant example of your written work, to:

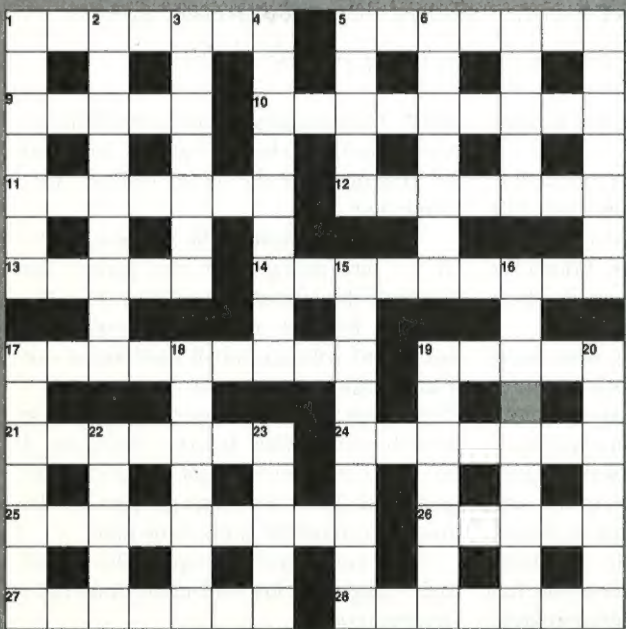
Will Watts, .EXE Magazine, Process Communications Ltd, 10 Barley Mow Passage, Chiswick, London W4 4PH

Editorial Index

If you are interested in the News articles on pages 6, 8 & 10, and would like to receive more information about the products mentioned, please circle the correspondong circle numbers on the Reader Service Card at the back of this magazine.

COMPANY	PRODUCT	CIRCLE NO
Ashton Tate	dBASE IV for Suns	816
BSI	QA Standard (BSI 7165)	817
Cocking & Drury	Smalltalk/V for Windows	818
ECUG	C++ User Group	819
Ergo	DOS Extender	820
EUROPAL	LISP Proceedings	821
Great Western Instruments	TDREM/LOCATE	822
Greenleaf	DataWindows	823
Gupta	SQLWindows for Btrieve	824
Loughborough University	Modula-2 Conference	825
Microsoft	SQL SGK/LADDR/MASM	826
Nu-Mega	NetWare Debuggers	827
Programming Research	QA C	828
QA Training	GPF/CASE:W	829
Roundhill	Periscope	830
Software Construction Co.	Grey Import Offer	831
Solution Systems	Brief V3.1	832
System C	Sycero C	833
System Science	Lahey FORTRAN	834
Systemstar	386 I Dos-Extender	835
TechnoJock Software	Object Toolkit	836
VRBA & Associates	R-Tech	837

MAY .EXEWORD



ACROSS
1 & 5 Not a real computer at all! (7, 7)
9 Variable name used for 1dn (5)
10 It counts in fractions (9)
11 Cats using pointers? (7)
12 Chawing round for a good interaction (7)
13 Follower of the Pharaohs (5)
14 ASCII 38 (9)
17 Avoids sub-routine statements? (9)
19 Observe the micro-electronic clock maybe (5)
21 Somehow I sail south-east with links (7)
24 Passing the time with a loop (7)
25 Capture or produce part of a transistor (9)
26 Harden in your hearing (5)
27 Dusts in confusion with natural folk (7)
28 Chunk of orange store (7)

DOWN
1 Discs in the library? (7)
2 Jumped back and changed range (9)
3 Tie slug round as it's least pretty (7)
4 Rural outlook of standard screen (9)
5 Copy 1999 onto a chip (5)
6 Robert Cook bears the message... (7)
7 ...with, behind, one upset town (2, 3)
8 Very angry, anger at last (7)
15 Circumvent strings of security measures (9)
16 Strange bit at true characteristic (9)
17 It seems foolish trick holds 25 (7)
18 How people work with hardware and software, for instance (7)
19 Having a break with 24 maybe (7)
20 Most eminent at the end of the range (7)
22 Got sick of beer d'you say? (5)
23 Visions of where 18 are (5)

'.EXEWORD' compiled by Eric Deeson

APRIL .EXEWORD

ADVERTISERS INDEX

STOB - The games we play

Object-oriented programming, AI and virtual reality techniques should make the arcade games of the future much more exciting for the player. And the program, predicts Ms Stob.

'Better get your skates on, Dave; you'll be on in a moment.'

Dave glanced through the CRT nonchalantly, then leaned back again, garbage-collecting his local heap on a chunk of static RAM.

'Neah. No worries. He hasn't got past Screen 3 yet. Neville'll sort him out when he tries to get through the whirlpool.'

'I wouldn't be so sure, Dave, if I were you. He got me before I had time to draw my laser.'

Dave looked at him sadly. 'With all due respect, Cyril, my son, *everybody* gets you before you have time to draw your laser. Even the old grannies who put their money in by mistake, thinking it's a fruit machine, and spend half the game trying to fire with the coin return button, even *they* get you before you have time to draw your laser.'

'Now, now, Dave,' said a third alien, waving an admonishing tentacle, 'you're forgetting that our Cyril here serves a purpose. He is our loss leader, as it were. Punter gets on machine, shoots Cyril to smithereens, thinks This is a piece of pisciculture, and spends the rest of the evening with us, barring the odd journey to the bar to crack a fresh £20 note for change.'

'I jolly well *could* give them a run for their money, if it wasn't for that wretched Iron Maiden pounding out all the time,' said Cyril, scratching his status bits. 'I don't see how anyone can be expected to shoot straight with that racket going on.'

'Oh I *am* sorry', said Dave, with heavy irony. 'Hi am sorry Hif the music is putting Sir off. I was under the impression that we were providing an all-action, machismo fantasy scenario, not some sort of girlie PacPerson thing for the ladies to toy with while their gentlemen drop bits of cheese-and-gherkin crisp into their dry white wines. Perhaps Sir would prefer it if we had a little discrete Jive Bunny-Rabbit, or maybe a sweet love-ballad by Andrew Lloyd-Web...'

A tremendous explosion rocked the machine. A tattered dragon, with three of its fire-breathing heads and its nuclear-missile launching tail shot off, staggered out of bit-mapped video RAM.

'Blimey, he's a sharp one,' said the dragon, reloading its missing pieces from ROM. 'Kept dodging behind the triffids before I could get a clear shot at him. We could be looking down the wrong end of a free game here.'

'Now that's defeatist talk, Bill', said Dave. 'If we start giving away free games, the takings'll drop faster than inflation in a Government forecast, and we'll be out in the back yard with a pin-ball machine in our place before you can say Power On Self Test. Anyway,' he continued, having taken a fresh peek at their human combatant, 'I still don't rate him. He's got his hand on his girlfriend's knee. You can't get past Neville *and* have a fumble at the same time.'

'Since you're such an expert, Dave,' said Bill, 'I might ask: just how many times have *you* been on.'

Dave check-summed his instance data, which is the way an artificially-intelligent object blushes. 'Well, obviously, coming in behind a fighter of Neville's calibre, I can't boast as much experience as...'

Once again, there was a large explosion. 'Blooming heck,' said Cyril, his pixels turning pale. 'He's shot Neville. You're on.'

EXE

If ... t of ... r ... ma... in it

Choosing your next compiler is no mean feat. Whether you're just putting a toe in the water or about to embark on a major development project with C++, you'll need to know that the compiler you choose gives you the productive edge.

TopSpeed C++ packs more in

TopSpeed C++ packs more in, whether you are developing for DOS, Windows 3 or OS/2. Develop mean-and-lean code with TopSpeed's super-optimizing code generator, Smart-Linking and Project system. Then, add the most extensive C++ class library available and you too can now harness the real power of C++.

TopSpeed Putting more into standards

TopSpeed C++ not only complies with the industry standards offering 100% conformance to AT&T V2.1, but also packs in a range of extensions to take your programming into hyperspace!

TopSpeed just the environment for C++

To make the most out of the C++ and the power of OOP you'll need the development environment that packs the most in. TopSpeed's unique Project system frees you to be creative whilst it manages file dependencies, compilation, assembly, linking and much more.

TopSpeed C++ in a family of thoroughbreds

C++ is just one of a family of four compilers that integrate seamlessly into the TopSpeed Multi-language Development Environment. From $99 per module you can custom-build a development system that takes your programming TopSpeed!

For more information call JPI on the number below.

EUROPE
+44 (0) 234 267500

CIRCLE NO. 808

JPI
Jensen & Partners International

USA
(415) 967 3200